Günter O. Hamann

BASIC

Schritt für Schritt mit

SHARP MZ-700

721 · 731 · 721 · 731 · 721 · 731 · 721 · 731 · 721 · 731 · 721 · 731 · 721 · 731 · 721 · 731 · 721 · 731 · 721 · 731 · 721 · 731

710 Seiten DM 29,80

Programmierte Unterweisung



DBV DEUTSCHER BETRIEBSWIRTE-VERLAG GMBH

Günter O. Hamann

BASIC
Schritt für Schritt mit
SHARP MZ-700

Programmierte Unterweisung

Programmierte Unterweisung

BASIC

Schritt für Schritt mit

SHARP MZ-700

Prof. Dr. Günter O. Hamann

DBV DEUTSCHER BETRIEBSWIRTE-VERLAG GMBH

Satz: GOHMOSS
Druck- und Bindearbeit: C.H. Beck'sche Buchdruckerei, Nördlingen
ISBN: 3-88640-014-X

VORWORT

Inhalt und Problemstellung

Für sehr viele Menschen ist es heute aus den unterschiedlichsten Gründen wünschenswert, über die Funktionsweise eines Computers informiert zu sein. Die angenehmste und sinnvollste Art, die vielfältigen Möglichkeiten dieser "Maschine" kennenzulernen und zu nutzen, besteht darin, deren Programmierung zu erlernen. Besonders geeignet ist hierfür das System SHARP MZ-700, weil es

- auch für den "Normalverbraucher" erschwinglich ist,
- sich durch ein besonders günstiges Preis-/Leistungsverhältnis auszeichnet,
- ein sehr zuverlässiges Gerät ist,
- nicht nur als "Hobby-Computer", sondern ebenso für echte "Profi-Anwendungen" verwendet werden kann.

Eine wichtige Programmiersprache dieses Rechners ist BASIC. Sie wurde bereits in den 60er Jahren von John Kemeny und Thomas Kurtz am Dartmouth College in Hanover / New Hampshire (USA) entwickelt und ist heute - ohne Hilfe von Industrie- oder Institutionenlobby und trotz aller kritischen Einwände - die verbreitetste Programmiersprache der Welt.

Die Ursache für diesen Erfolg ist wohl vor allem darin zu suchen, daß BASIC - im Gegensatz zu anderen Programmiersprachen - besonders schnell und leicht zu erlernen ist und somit vor allem dem Anfänger empfohlen werden kann.

Wie andere Computer-Sprachen auch, ist BASIC ständig weiterentwickelt worden. Dementsprechend erhöhte sich die Vokabelanzahl, wodurch zusätzliche Anwendungsmöglichkeiten entstanden. Hierbei ging jeder Hersteller - entgegen allen Normungsversuchen - zumindest teilweise eigene Wege, so daß es inzwischen eine Fülle von Dialekten gibt. Sie unterscheiden sich aber nur geringfügig, und die am häufigsten gebrauchten Vokabeln stimmen in fast allen Versionen überein. Für eine Einführung in BASIC ist es deshalb nicht wichtig, welche BASIC-Version man benutzt. Wichtig ist nur, daß man irgendein BASIC lernt.

Dann hat man die Voraussetzung, um eine etwas andere Fassung in kürzester Zeit zu durchschauen und zu verstehen.

Mit diesem Buch soll ein besonders mächtiges BASIC vermittelt werden, wie es auf dem System SHARP MZ-700 implementiert wurde. Damit haben wir die Gewähr, daß nahezu alle bei den verschiedenen Systemen der Praxis vorkommenden Befehle abgedeckt sind, die oft nur eine Untermenge des hier gebotenen Sprachumfangs zur Verfügung stellen.

Zielgruppe

Das Buch wendet sich vor allem an jene, die mit Hilfe eines Systems SHARP MZ-700 das Programmieren in BASIC lernen möchten.

Voraussetzungen

Diese **P**rogrammierte **U**nterweisung wurde konsequent für die Bedürfnisse eines Anfängers geschrieben. Zum Verständnis des Buches sind k e i n e fachspezifischen Vorkenntnisse erforderlich.

Lernziele

Nach dem Durcharbeiten aller Lektionen sollen die Adressaten in der Lage sein, selbständig

- BASIC-Programme für ein System SHARP MZ-700 zu erstellen,
- mit Hilfe der Handbücher und Systemunterlagen verschiedener Hersteller sich in kurzer Zeit in die BASIC-Versionen anderer Rechner einzuarbeiten.

Methode

Um den "Einstieg" zu erleichtern, wurde das Buch als **P**rogrammierte **U**nterweisung verfaßt: Der zu vermittelnde Stoff ist in kleine, überschaubare Lerneinheiten (**LE**) unterteilt. Jeder **LE** folgt eine Prüfeinheit (**PE**) mit Übungsaufgaben, Ergänzungs- und/oder Auswahltests, die unbedingt s c h r i f t l i c h zu lösen sind. Sie dienen sowohl der Erfolgskontrolle als auch der Festigung des Lernstoffes. Einer **PE** schließt sich die Lösung (**LÖS**) an, die jedoch niemals vor Bearbeitung der **PE** "konsultiert" werden soll. Weil es nicht ratsam

ist, die praktische Arbeit am Rechner erst dann aufzunehmen, wenn man sich umfangreiche Kenntnisse der Programmiersprache BASIC "erlesen" hat, findet der Leser ab Lektion 6 nach jedem Kapitel ein dokumentiertes **ÜBUNG**sprogramm, das dazu dient, den bis dahin gelernten Wissensstoff zu vertiefen und die Bedienung des Rechners kennenzulernen.

Unser Dank gebührt der Firma SHARP, insbesondere den Herren O. Eichler, M. Kaneko, K. Shimizu, für die freundliche Unterstützung.

Schortens, im November 1983 Günter O. Hamann

INHALTSVERZEICHNIS

Seite

Seite

Seite

Seite

Seite

Seite

Seite

1. Problemstellung

LE 1.1

Friesisches Tageblatt

1. 4. 1983

Die Alarich-Bande zittert vor SHARP MZ - 700

Mit Hilfe des SHARP-Computers MZ-700 gelang dem ostfriesischen Hobby-Programmierer G. O. H. (Name der Red. bekannt), was selbst den erfahrenen Spezialisten der Kriminalpolizei unmöglich war! Inzwischen weiß der SHARP MZ-700 mehr über das Verbrechen als alle Bandenmitglieder zusammen. -

Was geschah denn nun wirklich am Busento?

(Lesen Sie hierzu unseren ausführlichen Bericht auf Seite 7!)

Was verbirgt sich hinter dieser Notiz?

Der folgenden Darstellung sind die Hintergründe des "Falles" zu entnehmen:

Anno 1982 fuhren fünf Freunde aus Ostfriesland nach Italien, um am Ufer des Busento den Schatz des Alarich[1] zu suchen.

Sie gruben und suchten und suchten und gruben, um im Schweiße ihres Angesichts am Abend des 16. Juli erfolgreich zu sein: Nachdem sie viel Geröll beiseite geräumt hatten, entdeckten sie eine verfallene Truhe, bis zum Rande mit Solidi[2] gefüllt.

Es war zwar erst früher Abend - der Stundenzeiger der Uhr zeigte auf die 7; von des Tages Wühlarbeit waren sie jedoch so erschöpft, daß sie beschlossen, sich zunächst schlafen zu legen, um die gerechte Aufteilung des Fundes am nächsten Morgen vorzunehmen.

Sie fielen sofort in einen tiefen Schlaf, aber schon eine Stunde später erwachte der erste, von Mißtrauen gepeinigt, und beschloß, "aus Sicherheitsgründen" sich seinen fünften Teil zu nehmen. Um den Geist des Alarich zu beruhigen, warf er zunächst - entsprechend der Uhrzeit - 8 Solidi in den Busento, teilte dann die Münzen in fünf gleiche Teile (was genau "aufging"), versteckte seinen Anteil in seinem Gepäck und legte sich wieder schlafen.

Eine Stunde später erwachte der zweite. "Ich bin zwar ein guter Christ - aber die dort - na ja - ich nehme mir lieber meinen gerechten Teil", sagte er zu sich und schritt zur Tat. Entsprechend der Uhrzeit warf er zunächst 9 Solidi in den Busento, teilte dann die Goldstücke in fünf gleiche Teile (was genau "aufging"), versteckte seinen vermeintlichen Anteil in seinem Gepäck und legte sich schlafen.

Wieder eine Stunde später erwachte der dritte. Er verfuhr ähnlich wie seine Vorgänger: In Anlehnung an die Uhrzeit warf er 10 Solidi in den Busento, teilte die Goldstücke in fünf gleiche Teile (was genau "aufging"), versteckte seinen vermeintlichen Anteil und begab sich wieder zu Bett.

---

1) Alarich I, erster König der Westgoten, ist vor allem bekannt durch die Eroberung von Rom (24. Aug. 410 n. Chr.). Nach dreitägiger Plünderung der Stadt beabsichtigte Alarich, von Rhegium (= Reggio di Calabria) aus nach Sizilien und Afrika überzusetzen; während der Vorbereitungen dazu starb er noch im gleichen Jahr. Sein Leichnam wurde mit einem legendären Schatz bei Consentia (= Cosenza) im Flußbett des Busento begraben.

2) Der "Solidus" ist die goldene Standardmünze (4,55 g) des späten römischen Kaiserreiches, die um 324 n. Chr. von Constantin I für das gesamte Herrschaftsgebiet eingeführt wurde und bis zum Fall von Byzanz (1453) in Umlauf war.

Nach abermals einer Stunde erwachte der vierte - auch er warf entsprechend der Uhrzeit 11 Solidi in den Fluß, teilte dann die Goldstücke in fünf gleiche Teile (was genau "aufging"), versteckte seinen vermeintlichen Anteil in seinem Gepäck, um sich dann wieder schlafen zu legen.

Um Mitternacht erwachte der fünfte, der wie seine "Freunde" gemäß der Uhrzeit 12 Solidi zur Beruhigung des Alarich-Geistes in den Busento warf, um dann die Goldstücke in fünf gleiche Teile aufzuteilen (was abermals genau "aufging"), versteckte sein Fünftel im Gepäck und legte sich schlafen.

Nach Sonnenaufgang erwachten die Schatzgräber, machten alle ein ehrliches Gesicht, wie es nur Biedermännern eigen ist, und begannen mit der Aufteilung der restlichen Münzen, was genau "aufging".

Wieder in der Heimat angekommen, erfuhr die Polizei durch Indiskretionen von der Schatzräuberei, und wegen der Geschwätzigkeit der Ehefrauen konnte die "Verteilungsprozedur" exakt ermittelt werden.

Nur die Anzahl der gefundenen Münzen blieb unbekannt, und keiner der Beteiligten wollte sich diesbezüglich äußern. Die Kriminalpolizei fand schließlich den Hobby-Programmierer G. O. H., der mit Hilfe eines SHARP MZ-700 Computer-Systems berechnete, wie viele Goldstücke die Ostfriesen mindestens gefunden haben müssen.

PE 1.1

(1) Welche der folgenden Behauptungen ist/sind richtig?

A Drei Freunde fanden den Schatz des Alarich.

B Vier Freunde fanden den Schatz des Alarich.

C Fünf Freunde fanden den Schatz des Alarich.

D Der Schatz des Alarich bestand aus 573 Solidi.

E Der Schatz des Alarich bestand aus einer nicht direkt bekannten Anzahl von Solidi.

G Bei keiner der sechs Aufteilungen des Schatzes (bzw. des jeweiligen Restschatzes) verblieb ein Rest.

Der/die Lösungsbuchstabe/n (Lösungsbuchstabe = Buchstabe, neben dem eine richtige Behauptung aufgeführt ist) lautet/lauten

➧ C, E, G ....

(Bitte, tragen Sie den/die Lösungsbuchstaben an der gekennzeichneten Stelle ein!)

(2) (a)	Der erste erwachte um	. . . . .	Uhr
	und "opferte" dem Alarich-Geist	. . . . .	Solidi.
	Dann entwendete er heimlich	. . . . .	des Schatzes.
(b)	Der zweite erwachte um	. . . . .	Uhr
	und "opferte" dem Alarich-Geist	. . . . .	Solidi.
	Dann entwendete er heimlich	. . . . .	des Restschatzes.
(c)	Der dritte erwachte um	. . . . .	Uhr
	und "opferte" dem Alarich-Geist	. . . . .	Solidi.
	Dann entwendete er heimlich	. . . . .	des Restschatzes.
(d)	Der vierte erwachte um	. . . . .	Uhr
	und "opferte" dem Alarich-Geist	. . . . .	Solidi.
	Dann entwendete er heimlich	. . . . .	des Restschatzes.
(e)	Der fünfte erwachte um	. . . . .	Uhr
	und "opferte" dem Alarich-Geist	. . . . .	Solidi.
	Dann entwendete er heimlich	. . . . .	des Restschatzes.
(3)	Anläßlich der sechsten Aufteilung der verbliebenen Münzen am Morgen des nächsten Tages erhielt jeder der fünf "Freunde"	. . . . .	des Restschatzes.

LÖS 1.1

(1) Die Lösungsbuchstaben lauten
C , E , G .

(2)	(a)	Der erste erwachte um	8	Uhr
		und "opferte" dem Alarich-Geist	8	Solidi.
		Dann entwendete er heimlich	1/5	des Schatzes.
	(b)	Der zweite erwachte um	9	Uhr
		und "opferte" dem Alarich-Geist	9	Solidi.
		Dann entwendete er heimlich	1/5	des Restschatzes.
	(c)	Der dritte erwachte um	10	Uhr
		und "opferte" dem Alarich-Geist	10	Solidi.
		Dann entwendete er heimlich	1/5	des Restschatzes.
	(d)	Der vierte erwachte um	11	Uhr
		und "opferte" dem Alarich-Geist	11	Solidi.
		Dann entwendete er heimlich	1/5	des Restschatzes.
	(e)	Der fünfte erwachte um	12	Uhr
		und "opferte" dem Alarich-Geist	12	Solidi.
		Dann entwendete er heimlich	1/5	des Restschatzes.
(3)		Anläßlich der sechsten Aufteilung der verbliebenen Münzen am Morgen des nächsten Tages erhielt jeder der fünf "Freunde"	1/5	des Restschatzes.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 1.1** !

LE 1.2

Für die Beantwortung der Frage, wie viele Goldstücke die fünf Ostfriesen mindestens gefunden haben müssen, wollen wir unterstellen, daß wegen fehlender Voraussetzungen eine kurze mathematische Berechnung ausscheidet. Es drängt sich daher auf, die gesuchte Zahl - wie der Hobby-Programmierer G. O. H. - mit Hilfe eines SHARP MZ-700 Computer-Systems zu ermitteln.

Da dieses Buch für den Anfänger bestimmt ist und spezifische Vorkenntnisse nicht vorausgesetzt werden können, sind zunächst

- die Elemente eines Computer-Systems (Lektion 2),
- die Phasen der Programmerstellung (Lektion 3) und
- die Reihenfolgeplanung (Lektion 4)

zu behandeln, bevor wir Schritt für Schritt die Programmiersprache BASIC kennenlernen, die für unsere Problemlösung benötigt wird.

PE 1.2

Welche der folgenden Behauptung/en ist/sind richtig?

A Die Beantwortung der Frage, wie viele Goldstücke die fünf Ostfriesen mindestens gefunden haben müssen, soll aufgrund einer kurzen mathematischen Berechnung erfolgen.

B Da dieses Buch für den Anfänger bestimmt ist und spezifische Vorkenntnisse nicht vorausgesetzt werden können, sollen zunächst die Elemente eines Computer-Systems beschrieben werden.

C Nachdem wir die Elemente eines Computer-Systems vorgestellt haben, behandeln wir die Phasen der Programmerstellung, um uns anschließend der Reihenfolgeplanung zuzuwenden. Erst dann wollen wir uns Schritt für Schritt in die Programmiersprache BASIC einarbeiten.

D Da "BASIC" das Thema des Buches ist, müssen unverzüglich alle Vokabeln dieser Programmiersprache vorgestellt werden.

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

LÖS 1.2

Die Lösungsbuchstaben lauten

B, C .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 1.2** !

2. Elemente eines Computer-Systems

LE 2.1

In Lektion 1 (Problemstellung) sind wir übereingekommen, für die Lösung des "Alarich-Falls" ein Computer-System einzusetzen.

Was versteht man unter einem Computer-System?

Ein Computer-System ist eine Gesamtheit von sinnvoll zusammengestellten, interdependenten[1] Elementen, die man zur Durchführung der ADV[2] benötigt.

Zwei wichtige Aussagen können wir der obigen Definition entnehmen:

(1) Ein Computer-System besteht aus mehreren "Bausteinen" (= Elementen).

(2) Die "Bausteine" (= Elemente), aus denen ein Computer-System besteht, sind sinnvoll zusammengestellt und interdependent.

1) interdependent = gegenseitig voneinander abhängig

2) Wenn die Verarbeitung von Daten (= Informationen) mit Hilfe eines Computer-Systems vollzogen wird, dann spricht man von Automatisierter DatenVerarbeitung, abgekürzt ADV. Das Grundprinzip der ADV kann durch die Funktionen Eingabe, Verarbeitung, Ausgabe erklärt werden, nämlich der Eingabe, Verarbeitung und Ausgabe von Daten. Aus den Anfangsbuchstaben dieser Funktionen ergibt sich das Merkwort "EVA". - Sehr häufig verwendet man auch das Akronym EDV (= Elektronische DatenVerarbeitung), das wir für unzweckmäßig halten.

PE 2.1

Welche der folgenden Behauptungen ist/sind richtig?

A Ein Computer-System besteht aus einem Gerät, das alle Aufgaben ausführt, die im Rahmen der ADV anfallen.

B Ein Computer-System besteht aus mehreren "Bausteinen" (= Elementen).

C Die "Bausteine" (= Elemente), aus denen ein Computer-System besteht, müssen sinnvoll zusammengestellt sein. Außerdem existiert zwischen ihnen eine Interdependenz.

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

LÖS 2.1

Die Lösungsbuchstaben lauten

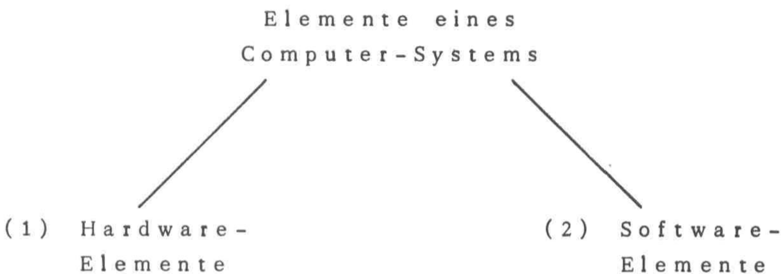
B, C .

Sollte Ihre Lösung nicht richtig sein, kehren Sie zurück zur
→ **LE 2.1** !

LE 2.2

Wir wissen nun, daß ein Computer-System aus einer Gesamtheit von sinnvoll zusammengestellten, interdependenten Elementen besteht, die man zur Durchführung der ADV benötigt.

Die Elemente eines Computer-Systems kann man in zwei große Gruppen aufgliedern:



Zu (1) Hardware-Elemente:

Hardware-Elemente sind vor allem die M a s c h i n e n eines Computer-Systems.

Zu (2) Software-Elemente:

Software-Elemente sind sowohl die D a t e n als auch die sog. P r o g r a m m e, durch welche die Arbeit der Hardware festgelegt wird.

PE 2.2

(1) Die Elemente eines Computer-Systems kann man in zwei große Gruppen aufgliedern:

. . . . und

. . . .

(2) Hardware-Elemente sind vor allem die

. . . .

eines Computer-Systems.

(3) Software-Elemente sind sowohl die Daten als auch die sog.

. . . .

durch welche die Arbeit der Hardware festgelegt wird.

LÖS 2.2

(1) Die Elemente eines Computer-Systems kann man in zwei große Gruppen aufgliedern:

H a r d w a r e - E l e m e n t e und

S o f t w a r e - E l e m e n t e .

(2) Hardware-Elemente sind vor allem die

M a s c h i n e n

eines Computer-Systems.

(3) Software-Elemente sind sowohl die Daten als auch die sog.

P r o g r a m m e,

durch welche die Arbeit der Hardware festgelegt wird.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 2.2** !

LE 2.3

Die Hardware-Elemente (=Maschinen) eines Computer-Systems bestehen aus:

(1) dem Rechner[1] und

(2) der sog. Peripherie[2] .

Zu (1): Der Rechner ist das wichtigste Gerät eines Computer-Systems. In ihm wird die eigentliche Datenverarbeitung vollzogen. Er besteht aus drei Baugruppen (vereinfachte Modellbetrachtung):

- Zentralprozessor,
- Hauptspeicher[3] und
- Ein-/Ausgabeprozessor.

Zu (2): Zur Peripherie eines Computer-Systems zählen wir nur jene Geräte, die on-line (=per Kabel) mit dem Rechner verbunden sind, z. B. Bildschirmgeräte, Drucker, Magnetbandgeräte und Magnetplattengeräte.

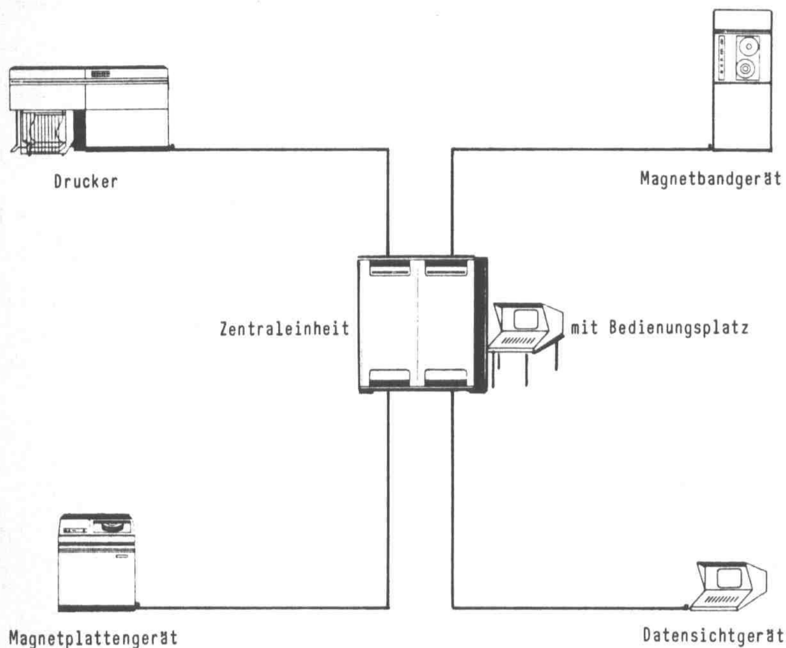
---

1) Man nennt den Rechner auch Computer (von engl. to compute = rechnen) oder Zentraleinheit.

2) Man spricht auch von "peripheren Geräten", d. h. Geräten, die um den Rechner herum aufgebaut sind.

3) Man bezeichnet den Hauptspeicher auch als Arbeitsspeicher.

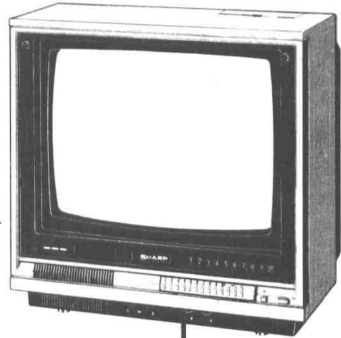
Beispiel für ein Computer-System:



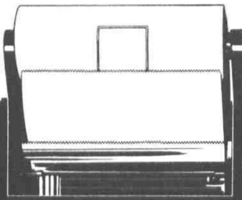
Beim Computer-System SHARP MZ-700 1) sind die Zentraleinheit mit Bedienungsplatz und die peripheren Geräte zu einer äußerst kompakten Anlage integriert:

Computer-System SHARP MZ-700 am Beispiel des MZ-731 mit ...

Datensichtgerät / Bedienungsplatz (in Form eines Fernsehers)



Plotter-Drucker



Bandlaufwerk

1) Wenn wir den Begriff SHARP MZ-700 verwenden, dann umfaßt er alle Modelle dieser Serie, vor allem den MZ-721 (Rechner mit eingebautem Bandlaufwerk) und den MZ-731 (Rechner mit eingebautem Plotter-Drucker und Bandlaufwerk).

PE 2.3

(1) Die Hardware-Elemente eines Computer-Systems bestehen aus

(a) .... und

(b) ....

(2) Das wichtigste Gerät eines Computer-Systems ist

....

(3) Der Rechner, in dem die eigentliche Daten v e r a r b e i t u n g vollzogen wird, besteht (bei vereinfachter Modellbetrachtung) aus den folgenden drei Baugruppen:

(a) .... ,

(b) .... ,

(c) ....

(4) Nennen Sie mindestens drei typische Peripheriegeräte!

(a) .... ,

(b) .... ,

(c) ....

LÖS 2.3

(1) Die Hardware-Elemente eines Computer-Systems bestehen aus

(a) dem Rechner und

(b) der Peripherie .

(2) Das wichtigste Gerät eines Computer-Systems ist

der Rechner .

(3) Der Rechner, in dem die eigentliche Daten**verarbeitung** vollzogen wird, besteht (bei vereinfachter Modellbetrachtung) aus den folgenden drei Baugruppen:

(a) Zentralprozessor ,

(b) Hauptspeicher ,

(c) Ein-/Ausgabeprozessor .

(4) Typische Peripheriegeräte sind

(a) Bildschirmgerät ,

(b) Drucker ,

(c) Magnetbandgerät ,

(d) Magnetplattengerät .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 2.3** !

LE 2.4

Wir haben gelernt, daß der Rechner das wichtigste Gerät eines Computer-Systems ist, denn in ihm wird die eigentliche Daten v e r - a r b e i t u n g vollzogen.

Die drei Baugruppen, aus denen ein Rechner besteht, haben folgende Aufgaben zu erfüllen:

(1) Zentralprozessor: Hier werden die Programmbefehle ausgeführt.

(2) Hauptspeicher: Hier werden
- Programme,
- zu verarbeitende Daten und
- Verarbeitungsergebnisse

gespeichert.

(3) Ein-/Ausgabeprozessor: Er regelt den Verkehr mit den angeschlossenen peripheren Geräten.

PE 2.4

In welchem Abschnitt (A oder B) werden die Aufgaben der Baugruppen eines Rechners richtig charakterisiert?

A Im Zentralprozessor werden die Programme, die zu verarbeitenden Daten und die Verarbeitungsergebnisse gespeichert. Der Hauptspeicher hingegen führt die Programmbefehle aus, während der Ein-/Ausgabeprozessor Eingabe- und Ausgabeoperationen verhindert, damit der Hauptspeicher ungestört arbeiten kann.

 → Seite 2-19

B Der Zentralprozessor führt die Programmbefehle aus. Im Hauptspeicher werden Programme, zu verarbeitende Daten und Verarbeitungsergebnisse gespeichert, und der Ein-/Ausgabeprozessor regelt den Verkehr mit den angeschlossenen peripheren Geräten.

 → Seite 2-18

(Bitte, eine Alternative ankreuzen!)

LÖS 2.4

Ihre Lösung ist richtig!

Denn die Programmbefehle werden im Zentralprozessor ausgeführt. Im Hauptspeicher - der Name verrät es schon - speichert das System die Programme, die zu verarbeitenden Daten und die Verarbeitungsergebnisse, und der Ein-/Ausgabeprozessor regelt den Verkehr mit den angeschlossenen peripheren Geräten.

→ Seite 2-21

LÖS 2.4

Ihre Lösung ist **nicht** richtig!

Sie sollten **LE 2.3** und **LE 2.4** nochmals aufmerksam durchlesen!

→ Seite 2-09

LE 2.5

Unterschiedliche Architektur gibt es nicht nur bei Gebäuden, sondern auch bei Rechnern.

In Abhängigkeit von der Rechnerarchitektur ist das Arbeiten mit einem Computer mehr oder weniger komfortabel. Für die Programmierung ist es heute wünschenswert, daß der Rechner *dialogfähig* ist.

Welche Eigenschaften sollte ein dialogfähiger Rechner besitzen?

(1) Sowohl die Eingabemöglichkeit für den Programmierer als auch die für den Anwender bestimmten Ausgaben des Rechners müssen in Griff- bzw. Sichtweite des Bedieners liegen.

(2) Es muß sowohl für den Bediener als auch für den Rechner möglich sein, Fragen/Aufgaben zu stellen und Antworten zu geben.

(3) Offensichtliche Fehler des Programmierers muß der Rechner melden, und zwar so genau lokalisiert, daß eine Korrektur leicht und schnell möglich ist.

Die obigen Eigenschaften besitzt das Computer-System SHARP MZ-700, auf dem wir unsere Beispielprogramme getestet haben.

PE 2.5

Welche der folgenden Eigenschaften sollte ein Rechner besitzen, den man als d i a l o g f ä h i g bezeichnet?

A Sowohl die Eingabemöglichkeit für den Programmierer als auch die für den Anwender bestimmten Ausgaben des Rechners müssen in Griff- bzw. Sichtweite des Bedieners liegen.

B Der Programmierer muß in Griffweite eine Eingabemöglichkeit besitzen; die Ausgaben des Rechners sollten hingegen nur auf dem Schnelldrucker eines Rechenzentrums erfolgen.

C Es muß sowohl für den Bediener als auch für den Rechner möglich sein, Fragen/Aufgaben zu stellen und Antworten zu geben.

D Es ist nicht zulässig, daß der Rechner Fragen stellt, weil das den Programmierer verwirren könnte, wodurch seine psychische Stabilität leiden müßte.

E Offensichtliche Fehler des Programmierers muß der Rechner melden, und zwar so genau lokalisiert, daß eine Korrektur leicht und schnell möglich ist.

F Auch offensichtliche Fehler darf der Rechner dem Programmierer nicht melden, weil sensible Menschen sich dadurch gekränkt fühlen könnten, wodurch die Freude am Programmieren leiden würde.

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

LÖS 2.5

Die Lösungsbuchstaben lauten

A, C, E .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 2.5** !

3. Phasen der Programmerstellung

LE 3.1

Wenn wir unseren Rechner sinnvoll einsetzen wollen, dann benötigen wir neben der Hardware noch ein

- Programm ,

das für einen von uns bestimmten Zweck erstellt wurde.

Was ist ein Programm?

In unserem Zusammenhang verstehen wir unter einem Programm nichts anderes als eine

- Gesamtheit von (Arbeits-)Anweisungen[1]

für den Computer.

1) Bedeutungsgleiche Begriffe für Anweisungen sind: Befehle, Instruktionen.

PE 3.1

(1) Für den sinnvollen Einsatz unseres Rechners benötigen wir neben der Hardware noch ein

Programm ....

(2) Unter einem Programm versteht man eine

Gesamtheit aller Anweisungen ....

für den Computer.

LÖS 3.1

(1) Für den sinnvollen Einsatz unseres Rechners benötigen wir neben der Hardware noch ein

P r o g r a m m .

(2) Unter einem Programm versteht man eine

G e s a m t h e i t v o n A n w e i s u n g e n

für den Computer.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 3.1** !

LE 3.2

Ein Programm setzt sich zusammen aus einer bestimmten Anzahl von (Arbeits-)Anweisungen für den Computer. Dabei müssen die Befehle selbstverständlich in einer sinnvollen Reihenfolge stehen. Zunächst ein

B e i s p i e l f ü r e i n f e h l e r h a f t e s P r o g r a m m:

1. Befehl: Gib drei beliebige Zahlen ein (in den Hauptspeicher des Computers)!

2. Befehl: Gib die Summe der soeben eingegebenen drei Zahlen aus (z. B. auf einen Bildschirm)!

3. Befehl: Addiere die drei Zahlen!

Warum ist das obige Programm fehlerhaft?

Die Summe kann logischerweise erst ausgegeben werden, n a c h d e m die eingegebenen Zahlen addiert worden sind.

Wenn wir die Reihenfolge der obigen Anweisungen ändern, dann erhalten wir ein

Beispiel für ein korrektes Programm:

1. Befehl: Gib drei beliebige Zahlen ein (in den Hauptspeicher des Computers)!

2. Befehl: Addiere die drei Zahlen!

3. Befehl: Gib die Summe der soeben eingegebenen drei Zahlen aus (z. B. auf einen Bildschirm)!

PE 3.2

Welche der folgenden Behauptungen (A oder B) ist richtig?

A Ein Programm besteht aus einer Anzahl von Anweisungen. Die Reihenfolge, in der die Anweisungen aufgeführt sind, ist unerheblich. Wesentlich ist vielmehr nur, daß die richtigen Befehle ausgewählt wurden. Der Computer sortiert sie und führt sie dann in der richtigen Reihenfolge aus.

☐ → Seite 3-08

B Ein Programm besteht aus einer bestimmten Anzahl von Anweisungen, die in einer sinnvollen Reihenfolge stehen müssen.

☒ → Seite 3-09

(Bitte, eine Alternative ankreuzen!)

LÖS 3.2

Ihre Lösung ist **nicht** richtig!

Zwar muß der Programmierer bei der Erstellung des Programms die "richtigen" Befehle erteilen; außerdem gewährt aber nur eine sinnvolle Reihenfolge der Anweisungen einen korrekten Programmlauf.

→ Seite 3-05

LÖS 3.2

Ihre Lösung ist richtig!

Denn nur die "richtigen" Befehle und ihre sinnvolle Reihenfolge geben die Gewähr für einen korrekten Programmlauf.

→ Seite 3-10

LE 3.3

Das Erstellen eines Programms ist eine Tätigkeit, die in sechs abgrenzbare Arbeitsschritte unterteilt werden kann. Diese abgrenzbaren Arbeitsschritte nennt man Programmierphasen.

1. Programmierphase: **A**ufgabendefinition.

Jede Programmiertätigkeit beginnt mit der präzisen Definition der Aufgabe.

2. Programmierphase: **R**eihenfolgeplanung.[1]

In diesem Abschnitt plant man die Struktur des Programms und die Reihenfolge der Befehlsschritte, und zwar erstellt man entweder ein Struktogramm oder einen Programmablaufplan.[1]

3. Programmierphase: **B**efehle schreiben (z. B. in BASIC).

In Anlehnung an das Struktogramm oder den Programmablaufplan schreibt man nun die eigentlichen Befehle (z. B. in der Programmiersprache BASIC) - ein Vorgang, den man auch Codierung nennt.

---

1) Die Erfahrungen haben gezeigt, daß der Anfänger vor allem Schwierigkeiten bei der Reihenfolgeplanung hat. Er sollte daher bei der Einarbeitung in diese Thematik die Vorzüge einer **P**rogrammierten **U**nterweisung nutzen. (Vgl. z. B. Hamann, G. O.: **P**rogrammierte **U**nterweisung: Logik der Programmierung - Strukturierte und Normierte Programmierung mit Programmablaufplänen und Struktogrammen, 2., erweiterte und verbesserte Auflage, Deutscher Betriebswirte-Verlag GmbH, Gernsbach 1983 und Taylorix Fachverlag, Stuttgart 1983

4. Programmierphase: **E**liminierung formaler und logischer Fehler.

Es kommt kaum jemals vor, daß ein Programm "auf Anhieb" richtig läuft. Es enthält zunächst Fehler, die in zwei Typen unterschieden werden können:

Die f o r m a l e n Fehler beinhalten Verstöße gegen die Regeln der Programmiersprache. Beispielsweise lautet der Ausgabebefehl in BASIC "PRINT". Wenn man statt dessen "PRINNT" (also mit zwei "N") schreibt, dann liegt ein Verstoß gegen die BASIC-Regeln vor. Es handelt sich um einen formalen Fehler.

Die l o g i s c h e n Fehler beinhalten Verstöße hinsichtlich der Art und der Reihenfolge der gewählten Befehle. Dadurch erhalten wir falsche Verarbeitungsergebnisse.

5. Programmierphase: **I**nstallation (= Inbetriebnahme).

Hier erfolgt die Übernahme der Programme durch das Rechenzentrum und/oder die Anwendung durch die Benutzer.

6. Programmierphase: **T**eiländerungen.

Mit der Übergabe des Programms ist die Arbeit des Programmierers nicht beendet. Fast jedes Software-Produkt unterliegt während seiner Lebensdauer zahlreichen Teiländerungen (zwecks Anpassung an veränderte Voraussetzungen, Verallgemeinerung der Lösung, Korrektur weiterer Fehler etc.). Bisweilen sind die Änderungswünsche so umfangreich, daß ein neues Programm geschrieben werden muß. Wir beginnen dann wieder mit der 1. Programmierphase.

Die soeben dargestellten sechs Programmierphasen lassen sich leicht einprägen mit Hilfe des Merkwortes

A R B E I T .

Es wird - wie wir bei aufmerksamem Lesen feststellen konnten - durch die Anfangsbuchstaben der sechs Programmierphasen gebildet.

Zur Verdeutlichung wiederholen wir die sechs Programmierphasen:

A ufgabendefinition,

R eihenfolgeplanung,

B efehle schreiben (z. B. in BASIC),

E liminierung formaler und logischer Fehler,

I nstallation (= Inbetriebnahme),

T eiländerungen .

In diesem Buch wollen wir vor allem Grundkenntnisse in der Programmiersprache BASIC vermitteln, auf die wir uns in den folgenden Lektionen konzentrieren werden.

PE 3.3

(Bitte, die linke Seite abdecken!)

(1) Das Erstellen eines Programms ist eine Tätigkeit, die in sechs abgrenzbare Arbeitsschritte unterteilt werden kann. Diese abgrenzbaren Arbeitsschritte, die man Programmierphasen nennt, lassen sich mit Hilfe eines Merkwortes einprägen.

Das Merkwort lautet:

ARBEIT . . . .

(2) Demnach unterscheiden wir bei der Programmerstellung die folgenden Programmierphasen:

A . . . . ,

R . . . . ,

B . . . . ,

E . . . . ,

I . . . . ,

T . . . .

LÖS 3.3

(1) Das Erstellen eines Programms ist eine Tätigkeit, die in sechs abgrenzbare Arbeitsschritte unterteilt werden kann. Diese abgrenzbaren Arbeitsschritte, die man Programmierphasen nennt, lassen sich mit Hilfe eines Merkwortes einprägen.

Das Merkwort lautet:

A R B E I T .

(2) Demnach unterscheiden wir bei der Programmerstellung die folgenden Programmierphasen:

A ufgabendefinition,

R eihenfolgeplanung,

B efehle schreiben (z. B. in BASIC),

E liminierung formaler und logischer Fehler,

I nstallation (= Inbetriebnahme),

T eiländerungen .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 3.3** !

4. Reihenfolgeplanung

LE 4.1

Wir haben gelernt, daß nach der genauen Aufgabendefinition (= 1. Programmierphase) die Reihenfolgeplanung beginnt. In dieser 2. Programmierphase erstellen wir ein Struktogramm oder einen Programmablaufplan.

Wozu dient ein Struktogramm oder ein Programmablaufplan?

Ein Struktogramm oder ein Programmablaufplan dient zur **F**ixierung der **R**eihenfolge von **A**nweisungen mit **N**ormierten **Z**eichen. Diese Definition läßt sich leicht einprägen mit Hilfe des Merkwortes

F R A N Z .

Es wird - wie wir oben sehen konnten - durch die Anfangsbuchstaben der wichtigen Definitionselemente gebildet. Zur Verdeutlichung wollen wir die Definition wiederholen:

Ein Struktogramm oder ein Programmablaufplan dient zur

Fixierung

der **R**eihenfolge

von **A**nweisungen

mit **N**ormierten

Zeichen .

Bei der **F**ixierung der **R**eihenfolge von **A**nweisungen mit **N**ormierten **Z**eichen hat der einzelne Programmierer einen großen Spielraum. Das Sprichwort "Viele Wege führen nach Rom" gilt sinngemäß auch für die Reihenfolgeplanung. Die Lösung e i n e s bestimmten Problems

kann durch viele verschiedene Wege (= Struktogramme bzw. Programmablaufpläne) in logisch korrekter Weise erfolgen.

Es muß allerdings erwähnt werden, daß der "planerische Freiheitsraum" für den Programmierer bei Programmablaufplänen größer ist als bei Struktogrammen. Anders ausgedrückt: Bei Verwendung von Struktogrammen wird der "planerische Freiheitsraum" (durch dieses Planungsinstrument) absichtlich eingeschränkt, und zwar mit dem Ziel, übersichtliche Programme zu erstellen.[1)]

1) Eine ausführliche Darstellung der Reihenfolgeplanung finden Sie in dem folgenden Buch: Haman, G. O.: **P**rogrammierte **U**nterweisung: Logik der Programmierung - Strukturierte und Normierte Programmierung mit Programmablaufplänen und Struktogrammen, 2., erweiterte und verbesserte Auflage, Deutscher Betriebswirte-Verlag GmbH, Gernsbach 1983 und Taylorix Fachverlag, Stuttgart 1983

PE 4.1

(1) Nach der genauen Aufgabendefinition (1. Programmierphase) beginnt die Reihenfolgeplanung. In dieser 2. Programmierphase erstellen wir

entweder ein . . . .

oder einen . . . .

(2) Ein Struktogramm oder ein Programmablaufplan dient zur

. . . .

der . . . .

von . . . .

mit . . . .

. . . .

(3) Bei Verwendung von Struktogrammen wird - anders als bei den Programmablaufplänen - der "planerische Freiheitsraum" des Programmierers absichtlich eingeschränkt, und zwar mit dem Ziel,

. . . . Programme zu erstellen.



(Hier eine für Programme anzustrebende Eigenschaft ergänzen!)

LÖS 4.1

(1) Nach der genauen Aufgabendefinition (= 1. Programmierphase) beginnt die Reihenfolgeplanung. In dieser 2. Programmierphase erstellen wir

entweder ein S t r u k t o g r a m m

oder einen P r o g r a m m a b l a u f p l a n .

(2) Ein Struktogramm oder ein Programmablaufplan dient zur

F i x i e r u n g

der **R** e i h e n f o l g e

von **A** n w e i s u n g e n

mit **N** o r m i e r t e n

Z e i c h e n .

(3) Bei Verwendung von Struktogrammen wird - anders als bei den Programmablaufplänen - der "planerische Freiheitsraum" des Programmierers absichtlich eingeschränkt, und zwar mit dem Ziel,

ü b e r s i c h t l i c h e Programme zu erstellen.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 4.1** !

LE 4.2

Wir haben nun die Voraussetzungen erarbeitet, um den "ALARICH-Fall" einer Lösung näher zu bringen. Doch bevor man die Befehle des Programms schreibt, ist

- die Aufgabe zu definieren und
- ein Struktogramm oder ein Programmablaufplan zu erstellen.

Aufgabendefinition ("Alarich-Fall"):

Es ist die niedrigstmögliche Ganzzahl (A) zu ermitteln, die folgende Bedingungen erfüllt:

(1) Von der um 8 verminderten Zahl wird 1/5 errechnet, das ganzzahlig sein muß, und von der Zahl abgezogen.

(2) Ausgangspunkt ist die gem. (1) ermittelte Zahl: Sie wird um 9 vermindert, dann 1/5 errechnet, das ganzzahlig sein muß, und ebenfalls abgezogen.

(3) Ausgangspunkt ist die gem. (2) ermittelte Zahl: Sie wird um 10 vermindert, dann 1/5 errechnet, das ganzzahlig sein muß, und ebenfalls abgezogen.

(4) Ausgangspunkt ist die gem. (3) ermittelte Zahl: Sie wird um 11 vermindert, dann 1/5 errechnet, das ganzzahlig sein muß, und ebenfalls abgezogen.

(5) Ausgangspunkt ist die gem. (4) ermittelte Zahl: Sie wird um 12 vermindert, dann 1/5 errechnet, das ganzzahlig sein muß, und ebenfalls abgezogen.

(6) Ausgangspunkt ist die gem. (5) ermittelte Zahl: Sie wird durch 5 geteilt. Das Ergebnis muß eine Ganzzahl sein.

Da ein Computer u. a. die Eigenschaft besitzt, schnell und zuverlässig zu rechnen, sollten wir diese für einen einfachen Lösungsalgorithmus[1] nutzen, nämlich dergestalt, daß wir den Inhalt eines Datenfeldes A, mit 1 beginnend, darauf überprüfen, ob der Wert die genannten Bedingungen (1 - 6) erfüllt. Sobald eine der Bedingungen nicht zutrifft, erhöhen wir A um 1 und beginnen erneut mit der Untersuchung. Dieser Vorgang ist so lange zu wiederholen, bis das System die gesuchte Zahl gefunden hat, die daraufhin ausgegeben werden soll.

Den soeben vorgeschlagenen Lösungsweg wollen wir wiederholen, allerdings

- in knappster Form,
- unter Beschränkung auf das Wesentliche,
- unter Beibehaltung einer sinnvollen Reihenfolge und
- unter Zuhilfenahme jener Zeichen, wie sie ein Programmierer auf der üblichen EDV-Zeichenschablone vorfindet.[2]

Also wir **F**ixieren die **R**eihenfolge von **A**nweisungen mit **N**ormierten **Z**eichen, mit anderen Worten:

Wir erstellen einen Programmablaufplan.[3]

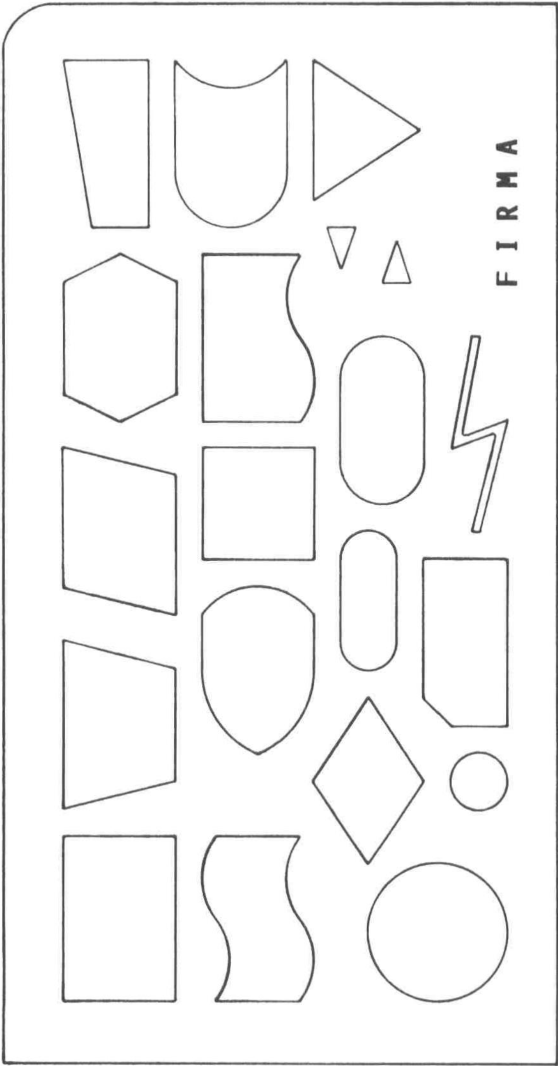
---

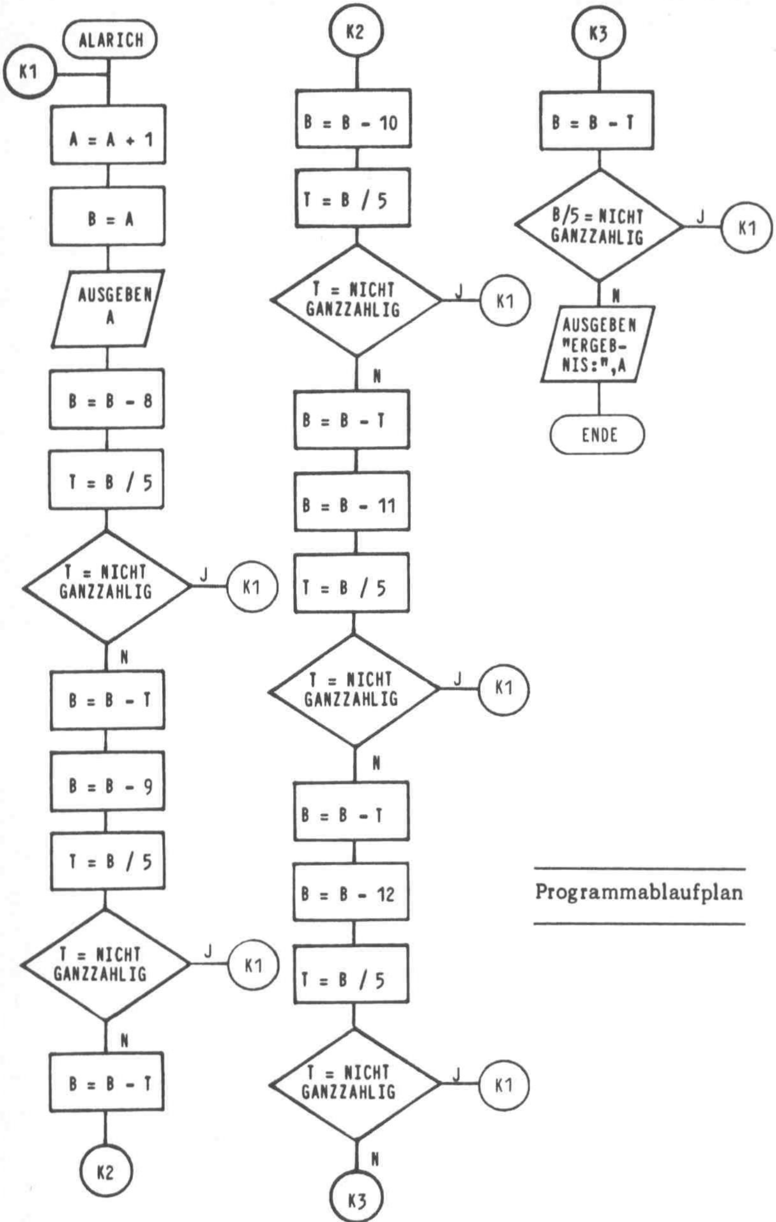
1) Unter einem Algorithmus versteht man einen nach einem bestimmten Schema ablaufenden Rechenvorgang. (Das Wort 'Algorithmus' verewigt den Namen des ABU JAFAR MUHAMMAD IBN MUSA AL-KHWARIZMI, der als Bibliothekar des Kalifen AL-MAMUN in Bagdad um das Jahr 825 ein Rechenbuch verfaßte, in dem die Methode des indisch-arabischen Rechnens - also mit Dezimalzahlen - dargestellt wurde. Die lateinische Übersetzung dieses Buches, welche um 1200 erfolgte, gibt den Verfassernamen etwas verfremdet mit 'AL GORITHMUS' wieder, um so eine Assoziation zum griechischen Wort 'arithmós' (= Zahl) zu erzielen.) Die Formulierung eines Algorithmus in einer Programmiersprache - z. B. BASIC - heißt 'Programm'.

2) Zum Zeichnen von Programmablaufplänen verwendet man zumeist eine Zeichenschablone nach DIN 66001 (vgl. folgende Seite), die im einschlägigen Fachhandel bezogen werden kann.

3) Da sowohl Struktogramme als auch Programmablaufpläne in der Praxis als Planungsinstrumente verwendet werden, wollen wir auch beide mit ihren Vorteilen und mit ihren Nachteilen anhand unseres Beispiels vorstellen. Weil die Reihenfolgeplanung mit einem Programmablaufplan das ältere Verfahren ist, bietet es sich an, daß wir damit beginnen.

Zeichenschablone nach DIN 66 001





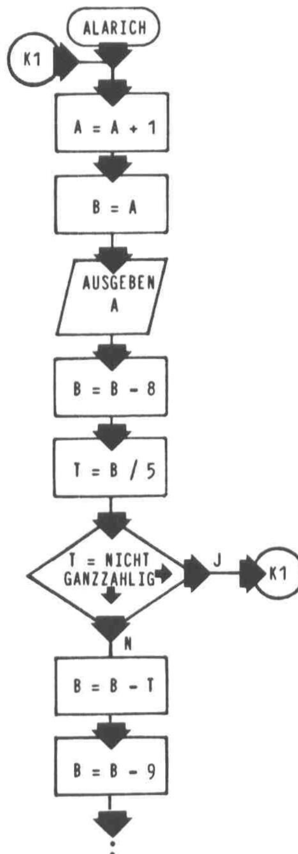
Programmablaufplan

Erläuterungen:

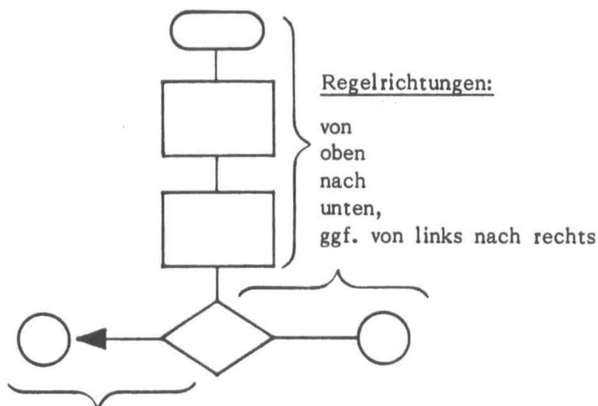
(1) Das Lesen eines Programmablaufplans (PAP) beginnt immer bei einem Zeichen, das "G r e n z s t e l l e" genannt wird. Es enthält meistens den Programmnamen oder ein Wort wie "ANFANG", "BEGINN", "START":

ALARICH

(2) Die regelmäßige Ablaufrichtung erfolgt von oben nach unten und von links nach rechts:



(3) Von der regelmäßigen Ablaufrichtung darf man in Ausnahmefällen abweichen. Die Abweichung muß jedoch immer durch eine Pfeilspitze hervorgehoben werden:



Hier wird ggf. von der Regelrichtung abgewichen, was durch eine Pfeilspitze hervorzuheben ist.

(4) Ein Programmablaufplan endet auch immer mit jenem Zeichen, das wir "Grenzstelle" nannten.

Allerdings enthält dieses Sinnbild am Ende eines Programmablaufplans das Wort "ENDE" oder "END":

ENDE

PE 4.2

(Bitte, die linke Seite abdecken!)

(1) In einem PAP erfolgt die regelmäßige Ablaufrichtung

von oben nach unten .... und

von rechts nach links ....

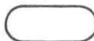
(2) Wird innerhalb eines PAP ausnahmsweise von der Regelrichtung abgewichen, muß dies durch eine

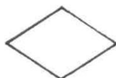
[illegible] ....

hervorgehoben werden.

(3) Welches der folgenden Sinnbilder wird als "Grenzstelle" bezeichnet und steht am Anfang und am Ende eines PAP?

 A

 B

 C

 D

Der Lösungsbuchstabe lautet

B ....

LÖS 4.2

(1) In einem PAP erfolgt die regelmäßige Ablaufrichtung

von oben nach unten und

von links nach rechts .

(2) Wird innerhalb eins PAP ausnahmsweise von der Regelrichtung abgewichen, muß dies durch eine

Pfeilspitze

hervorgehoben werden.

(3) Der Lösungsbuchstabe lautet

B .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 4.2** !

LE 4.3

Bei unserem PAP (vgl. folgende Seite) folgt nach dem Startsymbol (Grenzstelle mit dem Programmnamen "ALARICH") - wenn wir der vorgeschriebenen Ablaufrichtung folgen, also nicht von rechts nach links gehen - ein Rechteck, daß "Operation, allgemein" genannt wird:

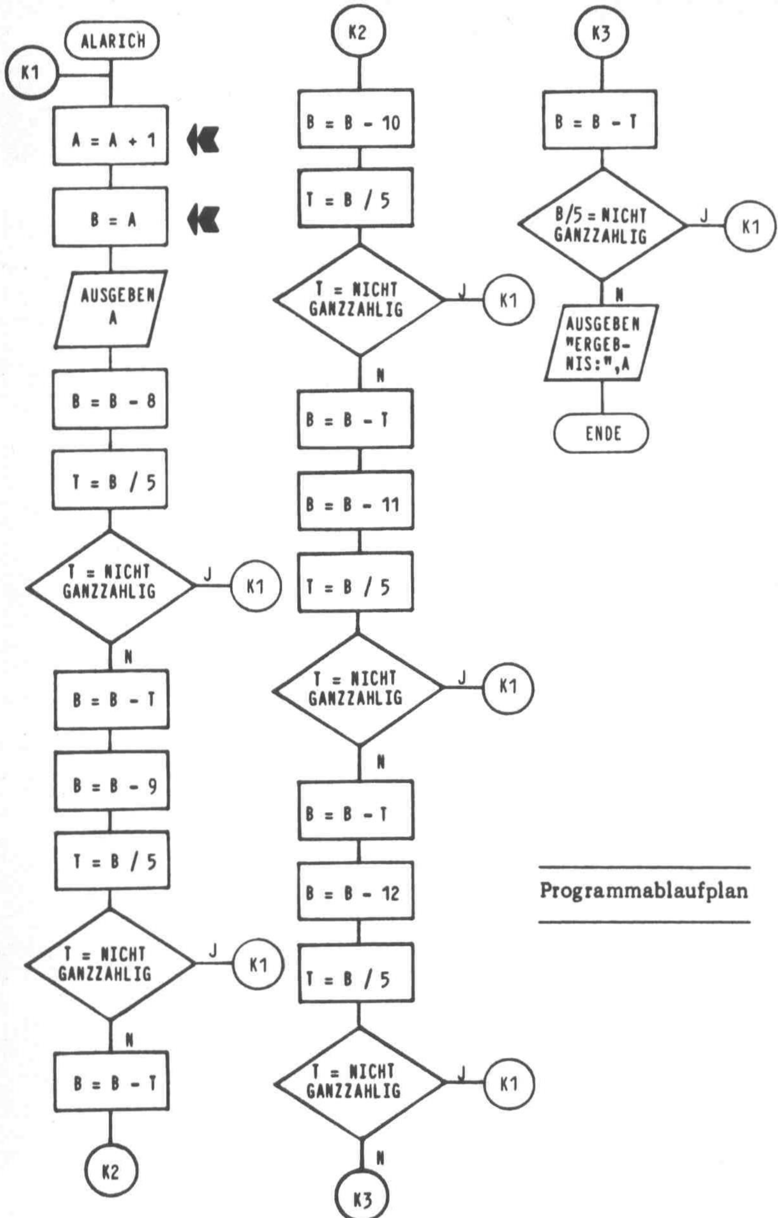
A = A + 1

Das Attribut "allgemein" wurde hinzugefügt, um den Gegensatz zu speziellen Operationen hervorzuheben, für die auch spezielle Zeichen gewählt wurden, die wir noch zu behandeln haben.

Das Rechteck, also das Symbol namens "Operation, allgemein", wird vor allem für arithmetische Operationen (Addition, Subtraktion etc.) eingesetzt.

Der Inhalt des angesprochenen Zeichens, also "A = A + 1", ist ein typischer Kurzeintrag, wobei es sich dabei nicht um eine mathematische Gleichung handelt. Das Gleichheitszeichen (=) ist hier vielmehr ein sog. Zuweisungszeichen. Dem Feld (auch "Variable" genannt) links vom Gleichheitszeichen soll der Wert des Ausdrucks zugewiesen werden, der rechts vom Gleichheitszeichen steht. Das Gleichheitszeichen bedeutet also nicht "ist gleich", sondern "ergibt sich aus" oder "(ist) soviel wie"; in unserem konkreten Fall: Der (neue) Wert von A ergibt sich aus dem (alten) Wert von A plus 1. Anders ausgedrückt: Erhöhe A um 1! (Hinweis: Bei BASIC-Programmen ist der Anfangswert von A automatisch = 0.)

Nach der Operation "A = A + 1" übertragen wir den Inhalt des Feldes A nach Feld B, und zwar mit dem Ziel, mit Hilfe von B die jeweilige Zahl definitionsgemäß zu verringern. Wenn dann die Überprüfung ergibt, daß eine der geforderten Bedingungen nicht erfüllt ist, haben wir in A noch den ursprünglichen, d. h. nicht verminderten Wert, der dann wieder um 1 zu erhöhen sein wird.



Programmablaufplan

Die Übertragung von A nach B läßt sich im PAP durch

B = A	(Inhalt von B ergibt sich aus dem Inhalt von A)

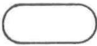
oder durch

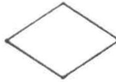
A ——► B	(Übertrage den Inhalt von A nach B!)

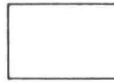
darstellen.

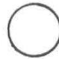
PE 4.3

(1) Welches der folgenden Sinnbilder wird "Operation, allgemein" genannt?


A


B


C


D

Der Lösungsbuchstabe lautet

....

(2) Welche der folgenden Behauptung/en ist/sind richtig?

A Man verwendet das Symbol "Operation, allgemein" vor allem für arithmetische Operationen und Zuweisungen (Übertragungen) im Hauptspeicher.

B Das Symbol "Operation, allgemein" verweist - wie der Name schon andeutet - auf eine Aktivität von allgemeiner Bedeutung, die deshalb mit EDV in keinem Zusammenhang stehen kann.

C Wenn das Symbol "Operation, allgemein" einen Eintrag mit einem Gleichheitszeichen enthält, so kann links davon nur ein Ergebnisfeld (Empfangsfeld) angegeben werden, dem der Wert des Ausdrucks (der Variablen) zuzuweisen ist, der (die) rechts vom Gleichheitszeichen steht.

Der/die Lösungsbuchstabe/n lautet/lauten

....

LÖS 4.3

(1) Der Lösungsbuchstabe lautet

C .

(2) Die Lösungsbuchstaben lauten

A , C .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 4.3** !

LE 4.4

Im Anschluß an die beiden erläuterten Operationen ("A = A + 1" und "B = A") folgt ein Parallelogramm (vgl. auch die folgende Seite!):



Der Name des Symbols, nämlich "Eingabe, Ausgabe", gibt schon Auskunft über seine Verwendung: Hiermit werden Eingabe- und Ausgabebefehle i. w. S. aufgeführt.

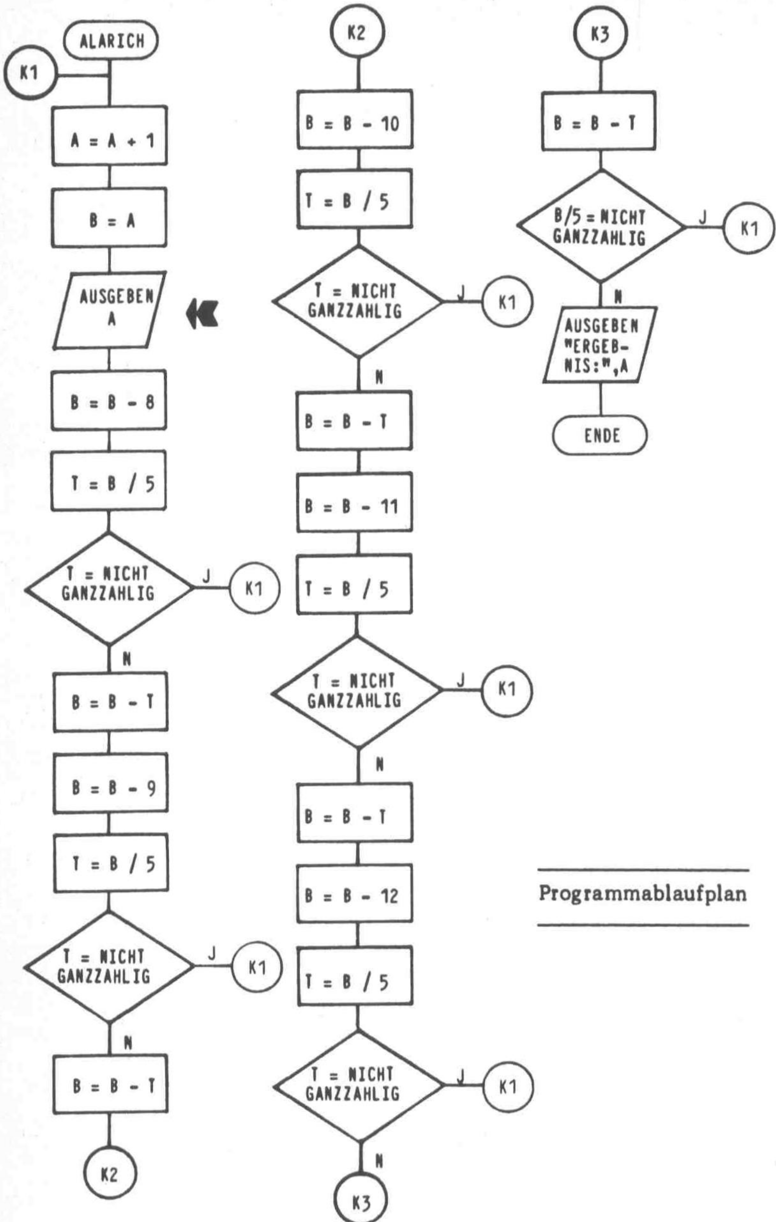
Zu den Eingabe- und Ausgabeanweisungen i. w. S. gehören vor allem

- Lesen (= Eingeben) von Datenfeldern,
- Lesen (= Eingeben) von Datensätzen,
- Eröffnen von Dateien[1],
- Schreiben (= Ausgeben) von Datenfeldern,
- Schreiben (= Ausgeben) von Datensätzen,
- Schließen von Dateien[1].

Ob es sich bei einer Instruktion um eine Eingabe oder Ausgabe handelt, ergibt sich immer "aus der Sicht des Hauptspeichers". Soll etwas in den Arbeitsspeicher gelangen, dann handelt es sich um eine Eingabe; hingegen nennt man den Transport von Informationen aus dem Hauptspeicher zu einem Peripheriegerät (z. B. Bildschirm, Drucker) eine Ausgabe.

---

1) Bevor der erste Datensatz einer Datei gelesen oder geschrieben werden kann, muß sie im Programm eröffnet worden sein. Und vor der Beendigung des Programms müssen alle Dateien geschlossen werden. (Da wir in unserer Aufgabe keine Datei anzusprechen haben, ist auch keine zu eröffnen und zu schließen.)

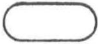
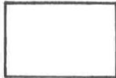


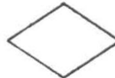
Programmablaufplan

In unserem konkreten Beispiel haben wir die Ausgabe des Inhalts von A vorgesehen. Für den eigentlichen Zweck des Programms benötigt man diese Anweisung nicht. Für den Anwender ist sie jedoch deshalb von Vorteil, weil er so während des Programmlaufs immer weiß, welcher Wert gerade untersucht wird.

PE 4.4

(1) Betrachten Sie die folgenden Zeichen!




A B C D E

Welches Sinnbild bezeichnet man mit "Operation, allgemein"? ....B....

Welches Sinnbild bezeichnet man mit "Eingabe, Ausgabe"? ....C....

Welches Sinnbild bezeichnet man mit "Grenzstelle"? ....A....

(Hier nur den jeweiligen Lösungsbuchstaben eintragen!)

(2) Bevor der erste Datensatz einer Datei gelesen oder geschrieben werden kann, muß man diese Datei

.eröffnet.werden.

(3) Vor Beendigung des Programmlaufs muß man die verwendeten Dateien

.geschlossen.... werden

(4) Für welche der folgenden Operationen verwendet man das Sinnbild mit dem Namen "Eingabe, Ausgabe"?

A Eröffnen von Dateien

B Lesen (= Eingeben) von Datenfeldern

C Addition von zwei Speicherfeldern

D Schließen von Dateien

E Lesen (= Eingeben) von Datensätzen

F Abfrage, ob ein numerisches Datenfeld einen positiven Inhalt hat

G Schreiben (= Ausgeben) von Datenfeldern

H Schreiben (= Ausgeben) von Datensätzen

Der/die Lösungsbuchstabe/n lautet/lauten

A, B, D, E, G, H . . . .

LÖS 4.4

(1) Das Sinnbild B bezeichnet man mit "Operation, allgemein".

Das Sinnbild C bezeichnet man mit "Eingabe, Ausgabe.

Das Sinnbild A bezeichnet man mit "Grenzstelle".

(2) Bevor der erste Datensatz einer Datei gelesen oder geschrieben werden kann, muß man diese Datei e r ö f f n e n .

(3) Vor Beendigung des Programmlaufs muß man die verwendeten Dateien s c h l i e ß e n .

(4) Die Lösungsbuchstaben lauten

A, B, D, E, G, H .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 4.2** !

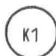
LE 4.5

Nachdem A ausgegeben worden ist, subtrahieren wir von B die Zahl 8, ermitteln dann 1/5 von B und speichern dieses Ergebnis in T. Im PAP (vgl. folgende Seite) schließt sich jetzt ein Zeichen an, das man "Verzweigung" nennt:



In Abhängigkeit von der hier gestellten Frage (T = NICHT GANZZAHLIG?[1]) fahren wir fort in Richtung "nach unten" oder "nach rechts".

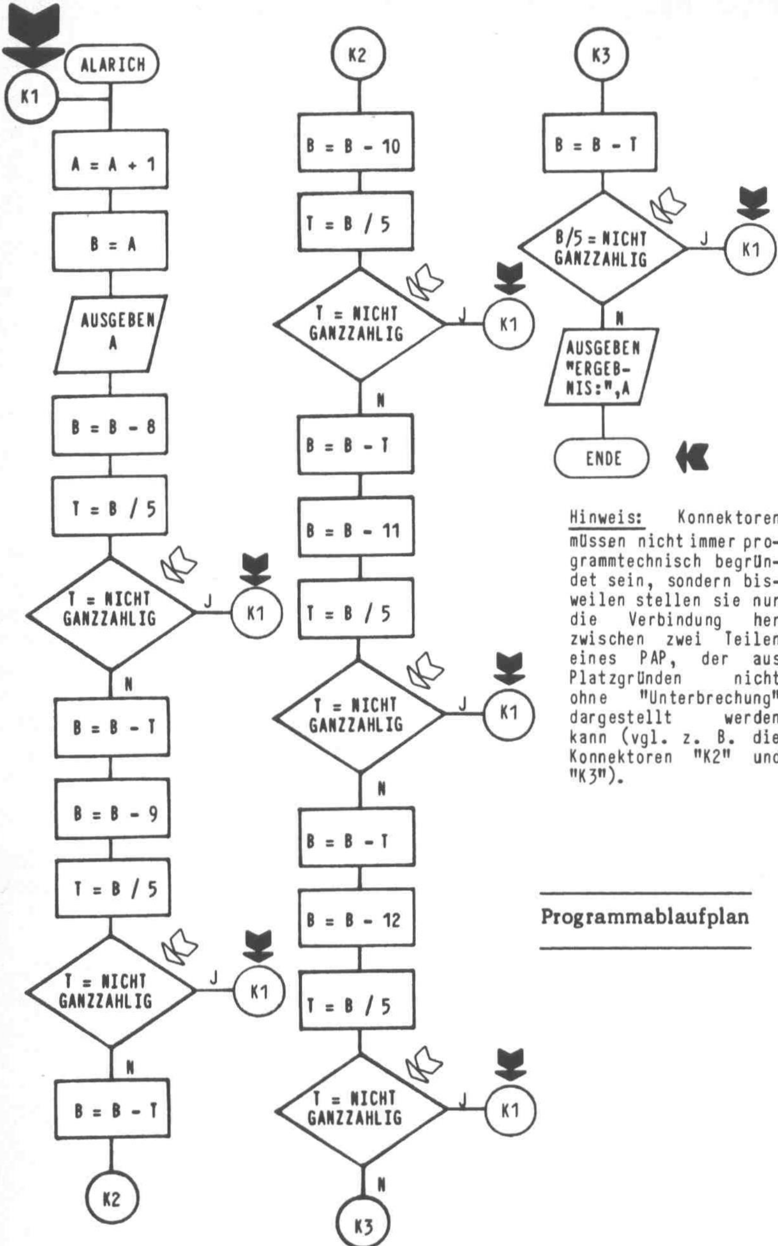
Die "Verzweigung" enthält immer eine Abfrage und hat immer mindestens zwei Ausgänge. Wenn die Frage "T = NICHT GANZZAHLIG?" mit "Ja" (= "J") beantwortet wirde, gelangen wir zu einem Zeichen, das man "Konnektor" oder "Übergangsstelle" zu nennen pflegt:



Wir müssen jetzt einen Konnektor suchen, der den gleichen Inhalt hat und bei dem - unter Einhaltung der Regelrichtung, also von oben nach unten bzw. von links nach rechts - weitere Zeichen folgen.

Ein bestimmter Konnektor, d. h. ein Konnektor mit festgelegtem Inhalt (z. B. "K1"), kann an mehreren Stellen als "Absprung" dienen. Er kann aber mit gleichem Inhalt nur an einer anderen Stelle als "Ansprung" verwendet werden. Andernfalls wäre der PAP nicht eindeutig, weil man nicht wüßte, wohin man springen soll.

1) Im Sinnbild braucht die Frage nicht mit einem Fragezeichen abgeschlossen zu werden, weil sich bereits aus dem grafischen Symbol der Raute ergibt, daß es sich um eine Frage handelt.



Hinweis: Konnektoren müssen nicht immer programmtechnisch begründet sein, sondern bisweilen stellen sie nur die Verbindung her zwischen zwei Teilen eines PAP, der aus Platzgründen nicht ohne "Unterbrechung" dargestellt werden kann (vgl. z. B. die Konnektoren "K2" und "K3").

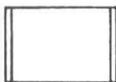
Programmablaufplan

Wir finden den zugehörigen A n sprungkonnektor unmittelbar unterhalb der Grenzstelle als Startsymbol.

Bei irgendeinem Wert von A, der uns nicht bekannt ist, wird das System die angesprochene Verzweigung mit dem Nein-Ausgang ("N") verlassen, um die Untersuchung fortzusetzen, bis schließlich irgendwann einmal alle sechs Verzweigungen mit dem Nein-Ausgang verlassen worden sind, und somit das gesuchte Ergebnis ausgegeben werden kann. Damit haben wir dann auch das Programmende erreicht:

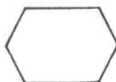
ENDE

Für das Erstellen von komplexen Programmablaufplänen benötigen wir noch das Zeichen "Unterprogramm" (= Rechteck mit zwei Zusatzstrichen), das in unserem PAP nicht aufgeführt ist:



Um die Übersichtlichkeit eines größeren Programmablaufplans zu gewährleisten, ist es möglich, mehrere sinnvoll zusammengehörige Operationen zu einem Unterprogramm zusammenzufassen. Die Details dieses Unterprogramms werden an anderer Stelle aufgeführt. Man nennt diese Vorgehensweise "Unterprogramm-Technik". Bei ihrer Anwendung entstehen ein Hauptprogramm (u. a. mit den Symbolen für Unterprogramme) und die (spezifizierten) Unterprogramme.

Auch das Sinnbild "Programm-Modifikation" haben wir in unserem sehr einfachen PAP nicht benötigt. Mit diesem Zeichen wird eine Operation hervorgehoben, wenn in Abhängigkeit von ihr zu einem späteren Zeitpunkt im Programmablauf eine Entscheidung getroffen werden soll:



Ein typischer Fall für die Verwendung der Programm-Modifikation ist das Setzen eines Schalters (auch "Merker", engl. "flag", genannt). Dabei bedeutet "1 ➔ Schalter" das Setzen eines Schalters auf "Ein". "0 ➔ Schalter" bedeutet das Setzen eines Schalters auf "Aus". In Abhängigkeit von der vorgefundenen Schalterstellung wird nach einer

Abfrage des Schalters an späterer Stelle des PAP der Ablauf des Programms mit unterschiedlichen Operationsfolgen fortgesetzt. Unter einem Schalter darf man sich allerdings keine technische Vorrichtung vorstellen. Er ist vielmehr nur ein kleines Speicherfeld, in das entsprechend der Absicht des Programmierers ausschließlich die Werte "1" oder "0" (d. h. die Schalterstellungen "Ein" oder "Aus") übertragen werden.

Nachdem wir bisher vor allem auf die Einzelheiten (= Elemente) eines Programmablaufplans eingegangen sind, wollen wir abschließend die wichtigsten Vor- und Nachteile dieses Verfahrens aufführen.

V o r t e i l e :

- großer Bekanntheitsgrad,
- keine erheblichen Schwierigkeiten bei Änderungen und Ergänzungen,
- keine wesentliche Einengung der planerischen Möglichkeiten.

N a c h t e i l e :

Es besteht heute weitgehende Einigkeit darüber, daß gute Programme eher dann entstehen, wenn man die Regeln der "Strukturierten Programmierung" (vgl. hierzu die folgende **LE 4.6** !) beachtet.

Es ist zwar möglich, auch beim Entwerfen eines PAP die Prinzipien der "Strukturierten Programmierung" zu befolgen. Die Einhaltung der Grundsätze ergibt sich aber nichts zwangsläufig. Vielmehr entsteht aufgrund des großen "planerischen Freiheitsraums" die Gefahr, daß der Programmierer

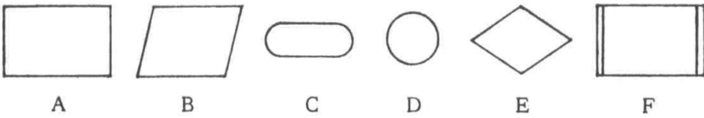
- unübersichtliche Programme

erstellt, die hinsichtlich Zuverlässigkeit, Wartungsfreundlichkeit, Kosten etc. negativ beurteilt werden müssen. Man spricht in diesem Zusammenhang treffend von "Spaghetti-Programmen".

PE 4.5

(1) Ordnen Sie den folgenden Sinnbildern die richtige Bezeichnung zu (durch Kombination der großen und kleinen Lösungsbuchstaben)!

Sinnbilder:



Bezeichnungen für Sinnbilder:

a "Unterprogramm"
b "Operation, allgemein"
c "Verzweigung"
d "Eingabe, Ausgabe"
e "Konnektor"
f "Grenzstelle"

Die richtigen Lösungsbuchstaben-Kombinationen lauten:

Buchstabe am Sinnbild:	Buchstabe an der Bezeichnung: (hier ergänzen!)
A	....
B	....
C	....
D	....
E	....
F	....

(2) Das Zeichen "Verzweigung" enthält immer

eine . . . .
und hat immer
(mindestens) zwei . . . .

(3) Ein Konnektor wird verwendet, um den Übergang von einer Anweisung zu einer anderen Anweisung darzustellen. Zusammengehörige Konnektoren müssen die gleiche Bezeichnung haben. Ein bestimmter Konnektor (z. B. "K1") kann

an mehreren Stellen als . . . .
dienen. Er kann aber mit gleichem Inhalt

nur an einer Stelle als . . . .
verwendet werden.

(4) Welche der folgenden Behauptungen ist richtig?

A Zur Wahrung der Übersichtlichkeit bei einem großen PAP können mehrere sinnvoll zusammengehörige Anweisungen zu einem Unterprogramm zusammengefaßt werden. Die Details des Unterprogramms werden an anderer Stelle spezifiziert.

B Das Zeichen "Unterprogramm" deutet an, daß dieser Teil des PAP bedeutungslos ist.

Der Lösungsbuchstabe lautet

A . . . .

(5) Welche der folgenden Behauptungen ist/sind richtig?

A Das Zeichen "Programm-Modifikation" verwendet man für arithmetische Operationen, weil damit eine Modifikation von Daten erfolgt.

B Das Zeichen "Programm-Modifikation" verwendet man zur Hervorhebung einer Operation, durch die der Ablauf bei einer späteren Verzweigung festgelegt wird.

C Das Zeichen "Programm-Modifikation" verwendet man insbesondere zum Setzen eines Schalters.

Der/die Lösungsbuchstabe/n lautet/lauten

B, C ....

(6) Unter einem Schalter darf man sich keine besondere technische Vorrichtung vorstellen. Er ist vielmehr nur ein kleines

.... ,

in das entsprechend der Absicht des Programmierers die Werte "1" oder "0" übertragen werden.

(7) Es ist zwar möglich, beim Entwerfen eines PAP die Prinzipien der "Strukturierten Programmierung" zu befolgen. Die Beachtung dieser Grundsätze ergibt sich aber nicht zwangsläufig. Vielmehr entsteht aufgrund des großen "planerischen Freiheitsraums" die Gefahr, daß der Programmierer

.... Programme erstellt.

(Hier eine negative Programmeigenschaft ergänzen!)

LÖS 4.5

(1) Die richtigen Lösungsbuchstaben-Kombinationen lauten:

Buchstabe am Sinnbild:	Buchstabe an der Bezeichnung:
A	b
B	d
C	f
D	e
E	c
F	a

(2) Das Zeichen "Verzweigung" enthält immer

eine Abfrage

und hat immer

(mindestens) zwei Ausgänge .

(3) Ein Konnektor wird verwendet, um den Übergang von einer Anweisung zu einer anderen Anweisung darzustellen. Zusammengehörige Konnektoren müssen die gleiche Bezeichnung haben. Ein bestimmter Konnektor (z. B. "K1") kann

an mehreren Stellen als Absprung

dienen. Er kann aber mit gleichem Inhalt

nur an einer Stelle als Ansprung

verwendet werden.

(4) Der Lösungsbuchstabe lautet

A .

(5) Die Lösungsbuchstaben lauten

B, C .

(6) Unter einem Schalter darf man sich keine besondere technische Vorrichtung vorstellen. Er ist vielmehr nur ein kleines

S p e i c h e r f e l d ,

in das entsprechend der Absicht des Programmierers die Werte "1" oder "0" übertragen werden.

(7) Es ist zwar möglich, beim Entwerfen eines PAP die Prinzipien der "Strukturierten Programmierung" zu befolgen. Die Beachtung dieser Grundsätze ergibt sich aber nicht zwangsläufig. Vielmehr entsteht aufgrund des großen "planerischen Freiheitsraums" die Gefahr, daß der Programmierer

u n ü b e r s i c h t l i c h e Programme erstellt.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 4.2** !

LE 4.6

Nachdem wir in den vorangegangenen Lerneinheiten die Reihenfolgeplanung am Beispiel des Programmablaufplans abgehandelt haben, wollen wir nun die Elemente eines Struktogramms erläutern. Hierbei wird es unerläßlich sein, zunächst kurz das Ziel und die wichtigste Maßnahme der "Strukturierten Programmierung" darzustellen, da sie die geistige Grundlage des zu beschreibenden Planungsinstruments bildet.

Zum besseren Verständnis erläutern wir die Zusammenhänge am Beispiel eines Struktogramms, das inhaltlich unserem PAP aus den vorangegangenen Lerneinheiten entspricht. Diese unmittelbare Umsetzung ist möglich, weil wir bei der Erstellung des Programmablaufplans die Grundsätze der "Strukturierten Programmierung" beachteten.

Neben den Unterschieden hinsichtlich des grafischen Erscheinungsbildes wollen wir schon jetzt hervorheben, daß bei Struktogrammen weniger auf die Art des Befehls (z. B. Eingabe oder arithmetische Operation oder Setzen eines Schalters etc.) abgestellt wird, als vielmehr auf die strukturelle Gegebenheit ("Befehle nur einmal ausführen", "Befehle mehrmals ausführen", "diese oder jene Befehle ausführen").

Ziel der "Strukturierten Programmierung" ist die Erstellung übersichtlicher Programme, um dadurch ihre

- Zuverlässigkeit,
- Wartungsfreundlichkeit,
- Entwicklungskosten etc.

zu verbessern.

Die wichtigste **Maßnahme** zur Erreichung dieses Ziels besteht darin, daß man den "planerischen Freiheitsraum" des Programmierers beschneidet. Anders ausgedrückt: Die Anzahl der logischen Konstruktionen, aus denen sich ein Programm zusammensetzen darf, wird standardisiert und auf ein Minimum beschränkt. (Anhand von Untersuchungen konnte nachgewiesen werden, daß dann gleichwohl jede programmtechnische Problemstellung lösbar bleibt.)

Nur die folgenden logischen Konstruktionen, auch **Strukturblockarten** genannt, sind zulässig:

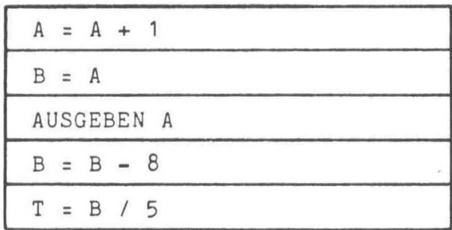
- Sequenz (= Folge),
- Alternation (= Auswahl),
- Iteration (= Wiederholung).

Eine **Sequenz** ist eine Folge von Operationen (Befehlen), die nacheinander ausgeführt werden.

```
A = A + 1
B = A
AUSGEBEN A
B = B - 8
T = B / 5
T = GANZZAHLIG
B = B - T
B = B - 9
T = B / 5
T = GANZZAHLIG
B = B - T
B = B - 10
T = B / 5
T = GANZZAHLIG
B = B - T
B = B - 11
T = B / 5
T = GANZZAHLIG
B = B - T
B = B - 12
T = B / 5
T = GANZZAHLIG
B = B - T
B / 5 = GANZZAHLIG
AUSGEBEN "ERGEBNIS:"; A
```

Beispiel einer S e q u e n z als

S t r u k t o g r a m m:



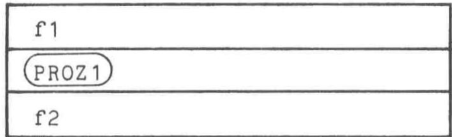
P A P:



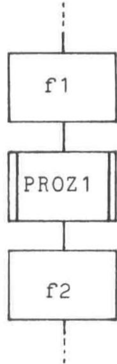
Eine Sonderform der Operation in einer Sequenz ist der Prozeduraufruf, durch den man eine ausgelagerte Befehlsfolge aufruft (Unterprogramm-Technik).

Beispiel eines P r o z e d u r a u f r u f s als

S t r u k t o g r a m m:



P A P:

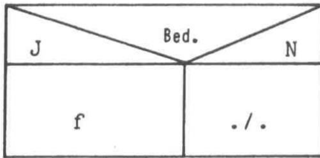


Alternation — Eine enthält (mindestens) eine bedingungsabhängige Verarbeitungsalternative.

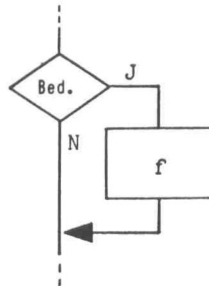
Man kann drei Unterfälle der Alternation unterscheiden.

(1) Bedingte Anweisung als

Struktogramm:

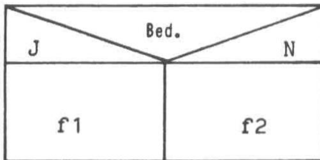


PAP:

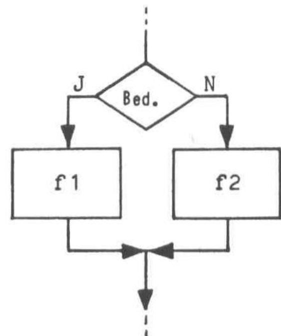


(2) Alternative Anweisung als

Struktogramm:

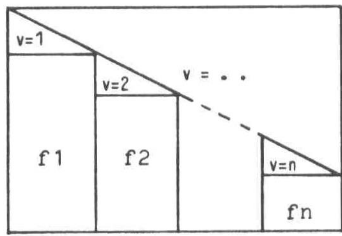


PAP:

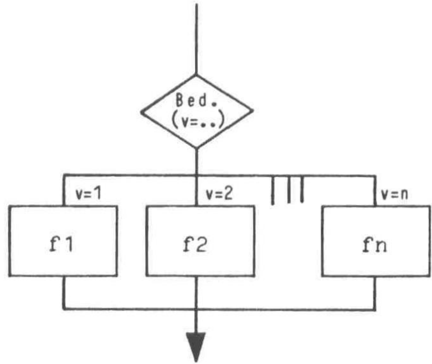


(3) Mehrfachverzweigung vorwärts (Fallunterscheidung) als

Struktogramm:



PAP:



In unserem einfachen Beispielprogramm haben wir keine Form der Alternation verwendet.

Die

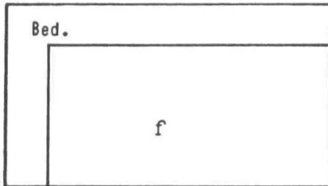
Iteration ist die dritte und letzte noch zu behandelnde Strukturblockart. Hierbei ist eine bestimmte Anzahl von Befehlen bedingungsabhängig mehrmals auszuführen.

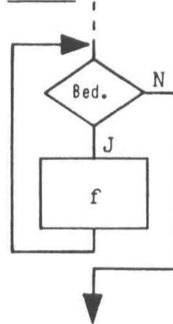
```
                        A = A + 1
                        B = A
                        AUSGEBEN A
                        B = B - 8
                        T = B / 5
                    T = GANZZAHLIG
                    B = B - T
                    B = B - 9
                    T = B / 5
                T = GANZZAHLIG
                B = B - T
                B = B - 10
                T = B / 5
            T = GANZZAHLIG
            B = B - T
            B = B - 11
            T = B / 5
        T = GANZZAHLIG
        B = B - T
        B = B - 12
        T = B / 5
    T = GANZZAHLIG
    B = B - T
B / 5 = GANZZAHLIG
AUSGEBEN "ERGEBNIS:"; A
```

Man kann drei Unterfälle der Iteration unterscheiden.

(1) Schleife vom Typ A[1] (=A-Schleife) als

Struktogramm: PAP:

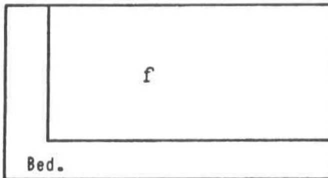


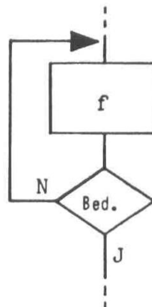


Hinweis: JA- und NEIN-Zweig sind festgelegte Elemente, auf die die Bedingung abgestimmt zu werden hat: Der JA-Zweig der Laufbedingung muß bei der A-Schleife in den Schleifenkörper führen.

(2) Schleife vom Typ Z[2] (=Z-Schleife) als

Struktogramm: PAP:





Hinweis: Der Schleifenkörper wird wenigstens einmal durchlaufen. Mit dem JA-Zweig der Bedingung ist der Schleifenkörper der Z-Schleife zu verlassen. - Wie wir dem nebenstehenden Struktogramm entnehmen können, enthält unsere Lösung mehrere geschachtelte Z-Schleifen.

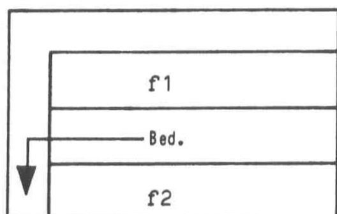
---

1) **A**m **A**nfang des Strukturblocks erfolgt der Test der Laufbedingung.

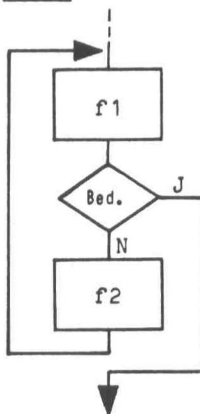
2) **Z**um Schluß erfolgt der Test der Laufbedingung.

(3) Schleife vom Typ M[1] (=M-Schleife) als

Struktogramm:



PAP:



Hinweis: Mit dem (den) JA-Zweig(en) der Bedingung(en) wird der Schleifenkörper der M-Schleife verlassen.

1) **M**ittendrin erfolgt mindestens ein Abbruchtest.

PE 4.6

(1) Das Ziel der "Strukturierten Programmierung" ist die Erstellung übersichtlicher Programme, um dadurch ihre Zuverlässigkeit, Wartungsfreundlichkeit, Entwicklungskosten etc. zu verbessern. Die wichtigste Maßnahme zur Erreichung dieses Ziels besteht darin, daß man den "planerischen Freiheitsraum" des Programmierers beschneidet, indem man die Anzahl der logischen Konstruktionen (= Strukturblockarten), aus denen sich ein Programm zusammensetzen darf, standardisiert und auf ein Minimum beschränkt. Wenn man einmal von den jeweiligen "Unterfällen" absieht, kann man drei Strukturblockarten unterscheiden, nämlich

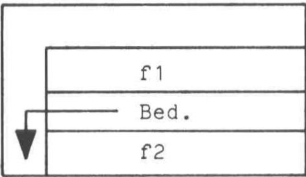
(a) .... ,

(b) .... ,

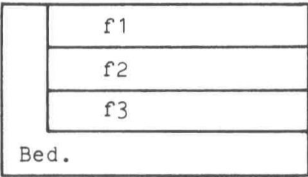
(c) ....

(2) Ordnen Sie den folgenden Strukturblockarten die richtige Bezeichnung zu (durch Kombination der großen und kleinen Lösungsbuchstaben)!

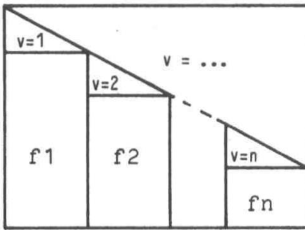
Strukturblockarten:



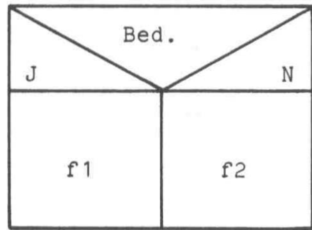
A



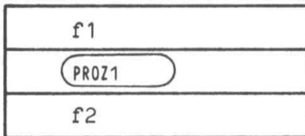
B



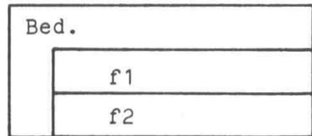
C



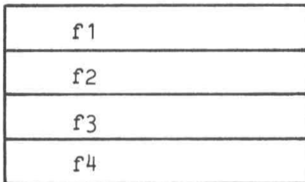
D



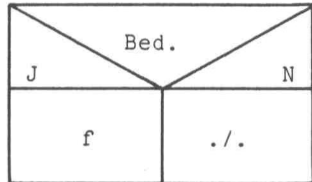
E



F



G



H

Bezeichnungen für Strukturblockarten:

a Sequenz (bestehend aus vier Operationen)

b Sequenz (bestehend aus Operation, Prozeduraufruf, Operation)

c Alternation in Form der bedingten Anweisung

d Alternation in Form der alternativen Anweisung

e Alternation in Form der Mehrfachverzweigung vorwärts (Fallunterscheidung)

f Iteration in Form der A-Schleife

g Iteration in Form der Z-Schleife

h Iteration in Form der M-Schleife

Buchstabe an der Strukturblockart:	Buchstabe an der Bezeichnung: (Hier ergänzen!)
A	. . h . . . .
B	. . . .
C	. . . .
D	. . d . . . .
E	. . b . . . .
F	. . f . . . .
G	. . a . . . .
H	. . c . . . .

(3) Welche der folgenden Behauptungen ist/sind richtig?

A Eine S e q u e n z ist eine Folge von Operationen (Befehlen), die nacheinander ausgeführt werden.

B Eine A l t e r n a t i o n enthält (mindestens) eine bedingungsabhängige Verarbeitungsalternative.

C Bei der A - S c h l e i f e sind JA- und NEIN-Zweig festgelegte Elemente, auf die die Bedingung abgestimmt werden muß: Der JA-Zweig der Laufbedingung muß bei der A-Schleife in den Schleifenkörper führen.

D Bei der A - S c h l e i f e verläßt man mit dem JA-Zweig den Strukturblock.

E Bei der Z-Schleife wird der Schleifenkörper wenigstens einmal durchlaufen. Mit dem JA-Ausgang der Bedingung ist der Schleifenkörper der Z-Schleife zu verlassen.

F Der Schleifenkörper der M-Schleife wird mit dem (den) JA-Zweig(en) der Bedingung(en) verlassen.

G Die M-Schleife enthält niemals eine Abbruchbedingung.

Der/die Lösungsbuchstabe/n lautet/lauten

A, B, C, E, F ....

LÖS 4.6

(1) Das Ziel der "Strukturierten Programmierung" ist die Erstellung übersichtlicher Programme, um dadurch ihre Zuverlässigkeit, Wartungsfreundlichkeit, Entwicklungskosten etc. zu verbessern. Die wichtigste Maßnahme zur Erreichung dieses Ziels besteht darin, daß man den "planerischen Freiheitsraum" des Programmierers beschneidet, indem man die Anzahl der logischen Konstruktionen (= Strukturblockarten), aus denen sich ein Programm zusammensetzen darf, standardisiert und auf ein Minimum beschränkt. Wenn man einmal von den jeweiligen "Unterfällen" absieht, kann man drei Strukturblockarten unterscheiden, nämlich

(a) Sequenz (= Folge) ,

(b) Alternation (= Auswahl) ,

(c) Iteration (= Wiederholung) .

(2) Die richtigen Lösungsbuchstaben-Kombinationen lauten:

Buchstabe an der Strukturblockart:	Buchstabe an der Bezeichnung:
A	h
B	g
C	e
D	d
E	b
F	f
G	a
H	c

(3) Die Lösungsbuchstaben lauten A, B, C, E, F .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 4.6** !

LE 4.7

In der vorangegangenen **LE 4.6** haben wir die Bausteine der "Strukturierten Programmierung" erläutert. Wir wollen nun die wichtigsten drei Regeln kennenlernen, die wir bei der Programmplanung mit Struktogrammen zu beachten haben.

1. Regel (Beschränkung der Strukturblockarten):

Die 1. Regel ist eine zusammenfassende Wiederholung: Bei der Konzeption eines Programms dürfen nur die Strukturblockarten Sequenz, Alternation und Iteration mit den festgelegten "Unterfällen" verwendet werden.

2. Regel (Top-Down-Konzept[1]):

Bei der Planung ist das sog. Top-Down-Konzept anzuwenden, d. h. die Lösung der Aufgabe erfolgt durch schrittweise Verfeinerung. Man beginnt also mit der Grobplanung und endet mit der Feinplanung. Konkret bedeutet dies, daß der Plan zunächst sehr viele Prozeduraufrufe enthält, die anschließend zu spezifizieren sind. (Nur bei sehr kleinen Programmen - wie unserem ersten Beispiel - ist eine schrittweise Verfeinerung nicht notwendig.)

3. Regel (Block-Konzept):

Strukturblöcke darf man nicht nur aneinanderreihen, sondern überdies schachteln, wie wir es in unserem Beispiel auch gemacht haben. Man erhält so "zusammengesetzte Strukturblöcke", bei denen grundsätzlich das Block-Konzept gewahrt bleiben muß. Das bedeutet:

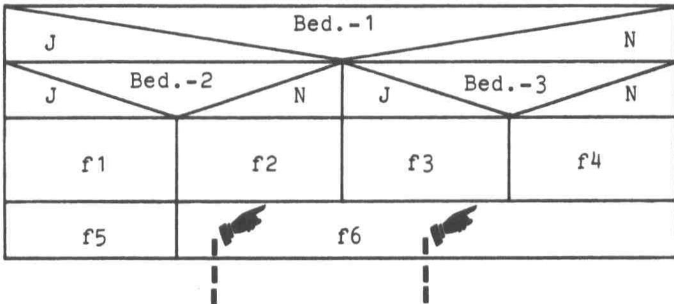
- Ein Strukturblock hat nur einen Eingang und nur einen Ausgang. (Die gesamte obere Kante ist der Eingang, und die gesamte untere Kante bildet den Ausgang.)

1) von engl. "top down" - Aussprache: top-daun - Übersetzung: von oben nach unten, d. h. von der hohen Abstraktionsebene zur niedrigen, also von der Grobplanung zur Feinplanung.

- Der Ausgang des voranstehenden Strukturblocks muß mit dem Eingang des folgenden Strukturblocks vollkommen übereinstimmen.

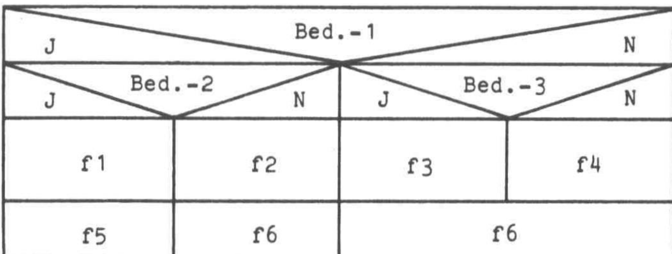
- Ein Strukturblock ist entweder vollständig in einem anderen Strukturblock enthalten oder enthält einen anderen Strukturblock (andere Strukturblöcke) vollständig. Teilweise Überlappung ist unzulässig.

Beispiel eines Struktogramms mit einem Verstoß gegen das Block-Konzept:



Der Block f6 hat zwei Eingänge, nämlich erstens im Anschluß zum Ausgang von f2 (NEIN-Zweig von Bed.-2) und zweitens im Anschluß an die alternative Anweisung mit Bed.-3. Eine solche Konstruktion ist nicht statthaft!

Korrigiertes Struktogramm ohne Verstoß gegen das Block-Konzept:



Bei konsequenter Planung mit Struktogrammen - unter Beachtung der Regeln der "Strukturierten Programmierung" - bestehen die

V o r t e i l e

darin, daß man

- übersichtliche Programme

erhält. Übersichtliche Programme zeichnen sich aus durch

- Zuverlässigkeit,
- Wartungsfreundlichkeit und
- geringe Kosten (bei Erstellung und Wartung).

Ein häufig verschwiegener

N a c h t e i l

der Struktogramme soll hier aber nicht unerwähnt bleiben: Die Durchführung von Änderungen und Ergänzungen (am Plan, nicht am Programm) ist vergleichsweise aufwendig (Schere und Klebe!).

PE 4.7

(1) Was versteht man unter "Beschränkung der Strukturblockarten" (1. Regel)?

(2) Erläutern Sie, was "Top-Down-Konzept" (2. Regel) bedeutet!

(3) Welches sind die richtigen drei Aussagen hinsichtlich des "Block-Konzepts" (3. Regel)?

A Ein Strukturblock hat nur einen Eingang und eine unterschiedliche Anzahl von Ausgängen.

B Ein Strukturblock hat nur einen Eingang und nur einen Ausgang. (Die gesamte obere Kante ist der Eingang, und die gesamte untere Kante bildet den Ausgang.)

C Der Ausgang des voranstehenden Strukturblocks muß mit dem Eingang des folgenden Strukturblocks vollkommen übereinstimmen.

D Der Ausgang des voranstehenden Strukturblocks wird mit dem Eingang eines beliebigen folgenden Strukturblocks mit Hilfe von zwei Konnektoren verbunden.

E Beim Schachteln von Strukturblöcken ist auch eine nur teilweise Überlappung zulässig.

F Ein Strukturblock ist entweder vollständig in einem anderen Strukturblock enthalten oder enthält einen anderen Strukturblock (andere Strukturblöcke) vollständig. Teilweise Überlappung ist unzulässig.

Die richtigen Lösungsbuchstaben lauten

B, C, F ....

LÖS 4.7

(1) Die Anzahl der logischen Konstruktionen, aus denen sich ein Programm zusammensetzen darf, wird standardisiert und auf ein Minimum beschränkt: Bei der Planung dürfen nur die Strukturblockarten S e q u e n z, A l t e r n a t i o n und I t e r a t i o n mit den festgelegten "Unterfällen" verwendet werden.

(2) Die Lösung einer Aufgabe erfolgt durch schrittweise Verfeinerung. Man beginnt also mit der Grobplanung und endet mit der Feinplanung.

(3) Die richtigen Lösungsbuchstaben lauten

B, C, F .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 4.6** !

Zur Lernsituation:

Es fällt zwar nicht übermäßig schwer, die in dieser Lektion behandelten einzelnen Elemente der Reihenfolgeplanung zu verstehen. Die Erfahrungen zeigen jedoch, daß der Gesamtzusammenhang von einem Anfänger nur mit großer Mühe durchschaut wird und auch die eigene praktische Anwendung große Schwierigkeiten und Probleme bereitet.

Wenn Sie daher den behandelten Problemkreis zunächst nicht voll erfaßt haben, braucht Sie das nicht zu beunruhigen. Durch unsere weiteren Ausführungen und die praktischen Programmierübungen in den folgenden Lektionen ergibt sich das Verständnis der wechselseitigen Zusammenhänge "von selbst".

5. Die Programmiersprache BASIC

LE 5

Wir haben gelernt, daß der Rechner ein Programm, also eine Gesamtheit von (Arbeits-) Anweisungen, benötigt, damit er Daten verarbeiten kann. Diese Programmbefehle können nicht in beliebiger Umgangssprache erteilt werden, da sie für die Maschine "Computer" zu kompliziert ist.

Andererseits ist es in der Regel auch nicht zumutbar, daß der "normale Computer-Anwender" die Maschinensprache des jeweiligen Rechners lernt. Sie ist meist sehr kompliziert und dementsprechend schwer zu erfassen. (Das gleiche gilt für die sog. maschinenorientierte Sprache, "Assembler" genannt.)

Folgende Lösung bietet sich an:

Man schafft eine stark vereinfachte Kunstsprache, wie z. B.

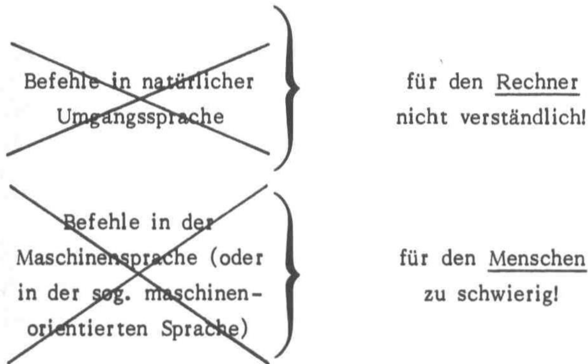
B A S I C, (Abkürzung für engl. "**B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode", frei übersetzt etwa "Symbolischer Allzweck-Befehlscode für Anfänger")

deren Sprachelemente der natürlichen englischen Sprache entlehnt sind und die deshalb vom Menschen leicht erlernt werden kann.

Der Hersteller des Rechners liefert diesen mit einem Übersetzungsprogramm[1], das die Befehle der vereinfachten Kunstsprache in Maschinenbefehle übersetzt, die dann vom Zentralprozessor "verstanden" und von ihm ausgeführt werden können.

1) Das Übersetzungsprogramm (BASIC-Interpreter) wird beim Computer-System SHARP MZ-700 auf einer Bandkassette (oder ggf. auf einer Diskette) geliefert.

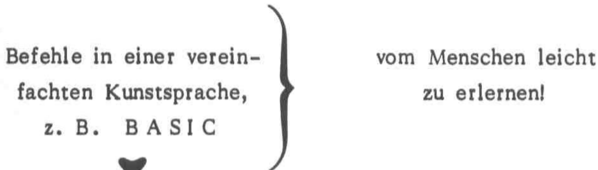
Schematische Zusammenfassung:



für den Rechner nicht verständlich!

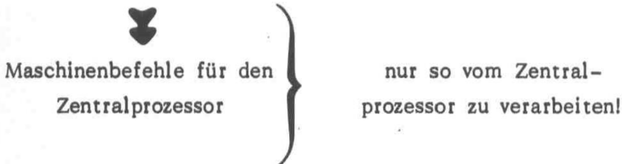
für den Menschen zu schwierig!

deshalb:



vom Menschen leicht zu erlernen!

Übersetzungsprogramm des Rechners (sog. Interpreter)



nur so vom Zentralprozessor zu verarbeiten!

PE 5

(Bitte, die linke Seite abdecken!)

Welche der folgenden Behauptungen ist/sind richtig?

A Rechner sind heute so anwenderfreundlich geworden, daß sie Programmbefehle in der natürlichen Umgangssprache verstehen.

B Die Maschinensprache des Rechners ist meist sehr kompliziert und dementsprechend schwer zu erfassen.

C BASIC ist eine vereinfachte Kunstsprache, deren Sprachelemente der natürlichen englischen Sprache entlehnt sind und die deshalb vom Menschen leicht erlernt werden kann.

D BASIC ist eine Kunstsprache. Sie ist schwerer zu erlernen als die Maschinensprache des Rechners.

E Die BASIC-Befehle müssen von einem Übersetzungsprogramm des Rechners in Maschinenbefehle übersetzt werden, denn nur sie kann der Zentralprozessor "verstehen" und ausführen.

Der/die Lösungsbuchstabe/n lautet/lauten

B, C, E ....

LÖS 5

Die Lösungsbuchstaben lauten

B, C, E .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 5** !

6. Programmbefehle und Systemkommandos

LE 6

Wenn man mit einem BASIC-Rechner arbeiten will, dann reicht es nicht aus, nur die Programmbefehle zu beherrschen. Darüber hinaus muß man auch die sog. "Systemkommandos" kennen. Demnach unterscheiden wir zwei Typen von Anweisungen:



Zu (1) Programmbefehle:

Sinnvoll zusammengesetzte Programmbefehle bilden das Anwenderprogramm. Programmbefehle sind - bei formaler Betrachtung - daran zu erkennen, daß ihnen Z e i l e n n u m m e r n vorangestellt sind. (Hierauf werden wir in **LE 7.1** näher eingehen.)

Zu (2) Systemkommandos:

Systemkommandos[1] gehören n i c h t zum Programm. Mit ihnen informiert man vielmehr den Rechner, was mit dem Programm oder Programmteilen geschehen soll. Rein formal betrachtet unterscheiden sich die Systemkommandos von den Programmbefehlen dadurch, daß den Systemkommandos im Regelfall keine Zeilennummern vorangestellt sind.

1) Sie sollten die folgenden Ausführungen zu den Systemkommandos zwar durchlesen. Es ist aber nicht unbedingt erforderlich, Details auswendig zu lernen, weil man sie sich müheloser im Rahmen der Dialogprogrammierung unserer ÜBUNGsprogramme einprägen kann.

Die folgenden Systemkommandos sind für die Arbeit mit einem SHARP MZ-700 unverzichtbar.

Systemkommandos:	**Erläuterungen:**
	Nachdem man eines der aufgeführten Kommandos mit der Tastatur geschrieben hat, wird es vom Rechner erst nach dem Drücken der [CR]-Taste ausgeführt.
LOAD	Sprachlicher Hinweis: von engl. "to load" = laden Funktion: Laden eines Programms vom Peripheriegerät in den Arbeitsspeicher, z. B. LOAD lädt das nächste Programm vom Kassettenband; LOAD "OTTO" sucht auf der Kassette ein Programm mit dem Namen "OTTO" und lädt es, wenn es gefunden wurde.
RUN	Sprachlicher Hinweis: von engl. "to run" = laufen Funktion: Übersetzung und Ausführung des im Arbeitsspeicher befindlichen Programms, z. B. RUN startet mit der Arbeit bei der niedrigsten Zeilennummer; RUN 120 startet ab Zeilennummer 120 .

Systemkommandos:	**Erläuterungen:**
SAVE	Sprachlicher Hinweis: von engl. "to save" = retten, sicherstellen Funktion: Speicherung des im Arbeitsspeicher befindlichen Programms auf einem Peripheriegerät, z. B. SAVE speichert das Programm auf dem Kassettenband; SAVE "OTTO" speichert das Programm auf dem Kassettenband mit dem Namen "OTTO". Achtung: Beim Speichern eines Programms auf Kassette prüft das System nicht, ob an der jeweiligen Stelle schon ein anderes Programm steht. Will man ein altes Programm nicht überschreiben und daher vor Erteilung des SAVE-Kommandos zunächst vorspulen, um exakt hinter dem alten Programm zu positionieren, so verwendet man hierfür die REWIND-Taste unter Beachtung des Bandzählwerks.
VERIFY	Sprachlicher Hinweis: von engl. "to verify" = verifizieren, auf Richtigkeit überprüfen Funktion: Überprüfung, ob das gespeicherte Programm mit dem aktuellen, im Arbeitsspeicher befindlichen Programm übereinstimmt. Hierdurch können fehlerhafte Aufzeichnungen aufgrund von mangelhaftem Bandmaterial, Funktionsstörungen etc. erkannt werden.

Systemkommandos:	Erläuterungen:

Beispiele:

VERIFY

überprüft das nächste Programm auf dem Kassettenband;

VERIFY "MAJA"

sucht auf der Kassette ein Programm mit dem Namen "MAJA" und vergleicht es nach dem Auffinden mit jenem im Arbeitsspeicher.

Anwendungsbeispiel:

Ein Programm "ALARICH" wurde mit SAVE gespeichert und das Band anschließend zurückgespult. Wenn man dann VERIFY mit der Tastatur geschrieben hat und die CR-Taste gedrückt worden ist, antwortet das System mit der Aufforderung ↓PLAY (= Drücke die PLAY-Taste auf dem Bandgerät!)

Nach Betätigung der PLAY-Taste meldet der Rechner schließlich

Found "ALARICH"

VERIFYING "ALARICH"

und schließlich

OK

Ready ,

d. h. die verglichenen Programme stimmen überein oder aber

READ error

Ready ,

d. h. das Programm ist nicht richtig abgespeichert, und das SAVE-Kommando ist zu wiederholen.

Systemkommandos:	**Erläuterungen:**
	Soll ein Programm auf Kassette überspielt werden, die beispielsweise schon zwei Programme enthält, so ist es bei einigen Systemen möglich, mit Hilfe von VERIFY eine Positionierung hinter dem zweiten Programm herbeizuführen, indem man zweimal vergleicht und zweimal eine Fehlermeldung ignoriert. Beim SHARP MZ-700 ist VERIFY hierfür n i c h t geeignet. Statt dessen ist die [REWIND]-Taste unter Beachtung des Bandzählwerks zu verwenden.
LIST	Sprachlicher Hinweis: von engl. "to list" = listen, auflisten Funktion: Auflisten des im Arbeitsspeicher befindlichen Programms (oder von Teilen davon), z. B. LIST bewirkt Auflistung des gesamten Programms; LIST 30 bewirkt Auflistung (nur) der Befehlszeile 30; LIST 30 - bewirkt Auflistung des Programms ab Zeile 30; LIST - 30 bewirkt Auflistung des Programmteils bis Zeile 30; LIST 30 - 90 bewirkt Auflistung des Programmteils von Zeile 30 bis Zeile 90. (Anm.: Das Auflisten kann man durch Niedergedrückthalten der Leertaste be-

Systemkommandos:	**Erläuterungen:**
	liebig lange unterbrechen und durch gleichzeitige Betätigung der Tasten [SHIFT] und [BREAK] beenden.)
DELETE	Sprachlicher Hinweis: von engl. "to delete" = streichen, tilgen Funktion: Löschen von Programmzeilen aus dem im Arbeitsspeicher befindlichen Programm, z. B. DELETE 30 bewirkt Löschung (nur) der Befehlszeile 30; DELETE - 30 bewirkt Löschung des Programmteils bis Zeile 30; DELETE 30 - bewirkt Löschung des Programms ab Zeile 30; DELETE 30 - 90 bewirkt Löschung des Programmteils von Zeile 30 - Zeile 90. (Anm.: Die Löschung einer konkreten Programmzeile kann auch damit bewirkt werden, daß man die Programmzeilennummer ohne irgendeinen Zusatz neu schreibt und durch Drücken der [CR]-Taste dem System übergibt.)
NEW	Sprachlicher Hinweis: von engl. "new" = neu, frisch Funktion: Löschung des im Arbeitsspeicher befindlichen Programms incl. der Datenfelder.

Systemkommandos:

Erläuterungen:

CONT

Sprachlicher Hinweis:

von engl. "to continue" = fortsetzen, fortfahren

Funktion:

Wurde ein Programmlauf aufgrund der Betätigung der Tasten [SHIFT] und [BREAK] oder aufgrund eines STOP- oder END-Befehls unterbrochen, dann setzt der Rechner die Arbeit an der "Unterbrechungsstelle" fort, wenn der Benutzer das Systemkommando CONT erteilt.

Steuercodes:

Steuercodes entsprechen in ihrer Wirkung den Systemkommandos. Man erteilt dem Rechner ein Kommando mittels Steuercode durch g l e i c h z e i t i g e s Drücken der [CTRL]-Taste und einer bestimmten Buchstabentaste. Beispielsweise bewirkt das g l e i c h z e i t i g e Drücken der [CTRL]-Taste und der [V]-Taste eine Löschung der auf dem Bildschirm wiedergegebenen Zeichen.

Ergänzender Hinweis: Bei Programmiersprachen, die wie SHARP-BASIC für "Interpreter-Betrieb" konzipiert wurden, ist die Unterteilung von Anweisungen in Programmbefehle einerseits und Systemkommandos andererseits nicht ohne Willkür, da hier letztlich nahezu jede Anweisung sowohl als Programmbefehl (mit Zeilennummer) als auch als Systemkommando (ohne Zeilennummer) verwendet werden kann. Die Zuordnung haben wir daher nach der üblichen Verwendungsart der Anweisung vorgenommen. (Wird das System mit einem Programm genutzt, spricht man auch von "indirekter Betriebsart" - die Anwendung einer Anweisung ohne Zeilennummer wird hingegen auch "direkte Betriebsart" genannt.)

Sie finden im Anhang-03 bis Anhang-20 eine ausführliche Zusammenstellung und Beschreibung sämtlicher Anweisungen, die üblicherweise als Systemkommando Verwendung finden und im Anhang-23 eine Erläuterung der Steuercodes.

PE 6

(1) Man unterscheidet zwei Typen von Anweisungen, nämlich Programmbefehle und

Systemkommandos ....

(2) Sinnvoll zusammengesetzte Programmbefehle bilden das Anwenderprogramm. Programmbefehle sind - bei formaler Betrachtung - daran zu erkennen, daß ihnen

Zeilennummern .... vorangestellt sind.

(3) Systemkommandos gehören n i c h t zum Programm. Mit ihnen informiert man vielmehr den Rechner, was mit dem Programm (oder Programmteilen) geschehen soll. Rein formal betrachtet unterscheiden sich die Systemkommandos von den Programmbefehlen dadurch, daß den Systemkommandos im Regelfall keine

Zeilennummern .... vorangestellt sind.

(4) Sie wollen ein auf Kassette gespeichertes Programm mit dem Namen "HAM" in den Arbeitsspeicher laden. Das entsprechende Systemkommando lautet:

Load "Ham" ....

(5) Der Rechner soll das im Arbeitsspeicher befindliche Programm übersetzen und ausführen. Das entsprechende Systemkommando lautet:

RUN ....

(6) Ein im Arbeitsspeicher befindliches Programm soll mit dem Namen "ALARICH" auf Kassette gespeichert werden. Das entsprechende Systemkommando lautet:

....

(7) Sie möchten sich von dem im Arbeitsspeicher befindlichen Programm die Zeilen 200 bis 300 auf dem Bildschirm auflisten lassen. Das entsprechende Systemkommando lautet:

LIST 200-300 ....

(8) Sie möchten das im Arbeitsspeicher befindliche Programm incl. der Datenfelder löschen. Das entsprechende Systemkommando lautet:

NEW ....

LÖS 6

(1) Man unterscheidet zwei Typen von Anweisungen, nämlich Programmbefehle und

Systemkommandos .

(2) Sinnvoll zusammengesetzte Programmbefehle bilden das Anwenderprogramm. Programmbefehle sind - bei formaler Betrachtung - daran zu erkennen, daß ihnen

Zeilennummern vorangestellt sind.

(3) Systemkommandos gehören n i c h t zum Programm. Mit ihnen informiert man vielmehr den Rechner, was mit dem Programm (oder Programmteilen) geschehen soll. Rein formal betrachtet unterscheiden sich die Systemkommandos von den Programmbefehlen dadurch, daß den Systemkommandos im Regelfall keine

Zeilennummern vorangestellt sind.

(4) Sie wollen ein auf Kassette gespeichertes Programm mit dem Namen "HAM" in den Arbeitsspeicher laden. Das entsprechende Systemkommando lautet:

LOAD "HAM" .

(5) Der Rechner soll das im Arbeitsspeicher befindliche Programm übersetzen und ausführen. Das entsprechende Systemkommando lautet:

RUN .

(6) Ein im Arbeitsspeicher befindliches Programm soll mit dem Namen "ALARICH" auf Kassette gespeichert werden. Das entsprechende Systemkommando lautet:

SAVE "ALARICH" .

(7) Sie möchten sich von dem im Arbeitsspeicher befindlichen Programm die Zeilen 200 bis 300 auf dem Bildschirm auflisten lassen. Das entsprechende Systemkommando lautet:

LIST 200 - 300 .

(8) Sie möchten das im Arbeitsspeicher befindliche Programm incl. der Datenfelder löschen. Das entsprechende Systemkommando lautet:

NEW .

Sollten Ihre Lösungen nicht richtig sein, ist es trotzdem nicht unbedingt erforderlich, die **LE 6** nochmals durchzuarbeiten. Informieren Sie sich besser im Bedarfsfall!

ÜBUNGsprogramm zu Lektion 6

(1) Hinweise zu den **ÜBUNG**sprogrammen:

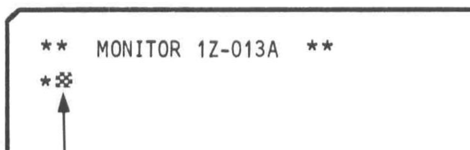
Es ist nicht ratsam, die praktische Arbeit am Rechner erst dann aufzunehmen, wenn man umfangreiche Kenntnisse der Programmiersprache BASIC erworben hat. Wir halten es für sinnvoller, schon hier mit dem Programmieren zu beginnen, selbst wenn bestimmte Einzelheiten noch nicht verstanden werden und u. U. einige unerwartete Schwierigkeiten aufgrund von Fehlbedienungen auftreten.

Deshalb finden Sie von jetzt ab im Anschluß an jede Lektion ein **ÜBUNG**sprogramm, das Sie nur zu übernehmen brauchen. Es soll dazu beitragen,

- den bis dahin gelernten Wissensstoff zu festigen,
- neue und unbekannte Befehle kurz kennenzulernen und ihre Wirkungsweise zu erfahren, obwohl Einzelheiten dazu erst an späterer Stelle vermittelt werden (Hiermit ist außerdem die Anregung verbunden, sich gelegentlich "im Vorwege" zu informieren und das Buch unter Verwendung des Stichwortverzeichnisses auch als "Nachschlagwerk" zu benutzen!),
- die Bedienung des Computer-Systems SHARP MZ-700 zu üben.

(2) Inbetriebnahme des Rechners:

Wir gehen davon aus, daß Sie den Rechner mit Hilfe der Bedienungsanleitung des Herstellers an einen Farbfernseher korrekt angeschlossen und angeschaltet haben. Auf dem Bildschirm erscheint dann die folgende Nachricht:



Der Cursor (Aussprache: "Körsser") - das ist das blinkende Zeichen - zeigt dem Anwender immer die aktuelle Schreibposition.

Hierdurch erfahren wir, daß der Monitor (= das Steuerprogramm) aktiv ist. Er ist in einem sog. Festwertspeicher, einem Teilbereich des Hauptspeichers, enthalten und braucht - im Gegensatz zu allen anderen Programmen - nicht mehr "von außen" in den Speicher übertragen zu werden.

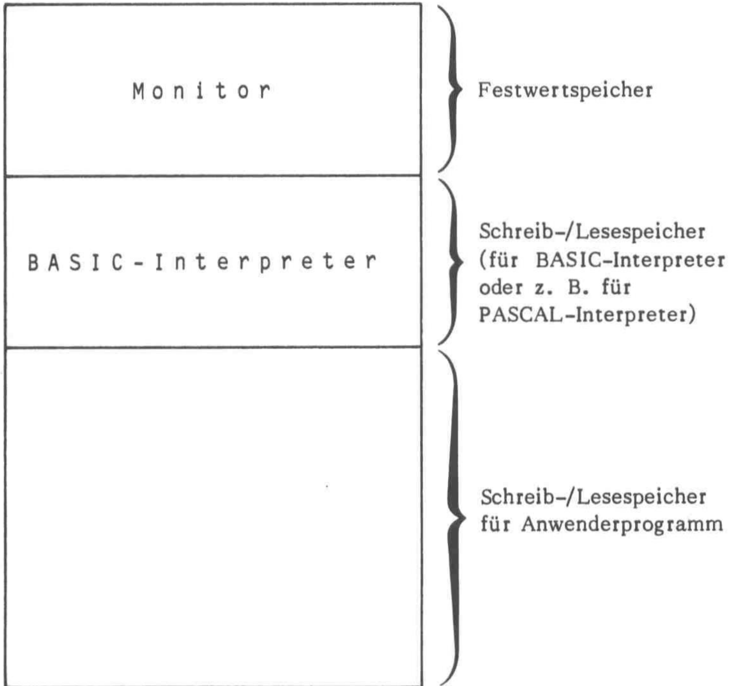
(3) Laden des BASIC-Interpreters:

Der BASIC-Interpreter ist ein Programm, das dazu dient, in der Programmiersprache BASIC abgefaßte Anwenderprogramme für den Zentralprozessor zu übersetzen und von diesem ausführen zu lassen. Im Gegensatz zum Monitor muß man den BASIC-Interpreter erst "von außen", d. h. von der Bandkassette, in den Hauptspeicher übertragen. Man bezeichnet diesen Vorgang auch als "L a d e n des BASIC-Interpreters". Hierdurch unterscheidet sich das Computer-System SHARP MZ-700 von vielen anderen Rechnern und bietet damit den Vorteil, daß z. B. wahlweise statt des BASIC-Interpreters in denselben Speicherbereich ein PASCAL-Interpreter geladen werden kann.

Vorgehensweise:

(a) Drücken Sie auf die STOP/EJECT -Taste des Bandkassettenlaufwerks! Hierdurch öffnet sich der Deckel des Geräts.

Aufteilung des Hauptspeichers (vereinfachtes Modell):



(b) Legen Sie die Bandkassette, die den BASIC-Interpreter enthält, dergestalt in die Deckelführung des Geräts, daß

- das Etikett nach oben liegt,
- das Bandmaterial sich auf der linken Spule befindet,
- Sie auf die Schreib-/Leseöffnungen der Bandkassette blicken können.

(c) Schließen Sie den Deckel mit der Bandkassette!

(d) Schreiben Sie mit Hilfe der Tastatur das folgende Kommando:

LOAD (Es reicht auch der einzelne Buchstabe L [1].)

und drücken Sie dann auf die CR-Taste[2]. Mit der Betätigung der CR-Taste übergibt man dem System das Kommando "LOAD".

(e) Der Rechner antwortet auf dem Bildschirm mit der Nachricht

↓ PLAY

und bringt damit zum Ausdruck, daß nun die PLAY-Taste des Kassettenlaufwerks zu drücken ist.

(f) Nach Betätigung der PLAY-Taste beginnt der Ladevorgang, was durch das System kurze Zeit später mit folgender Nachricht bestätigt wird:

LOADING S-BASIC

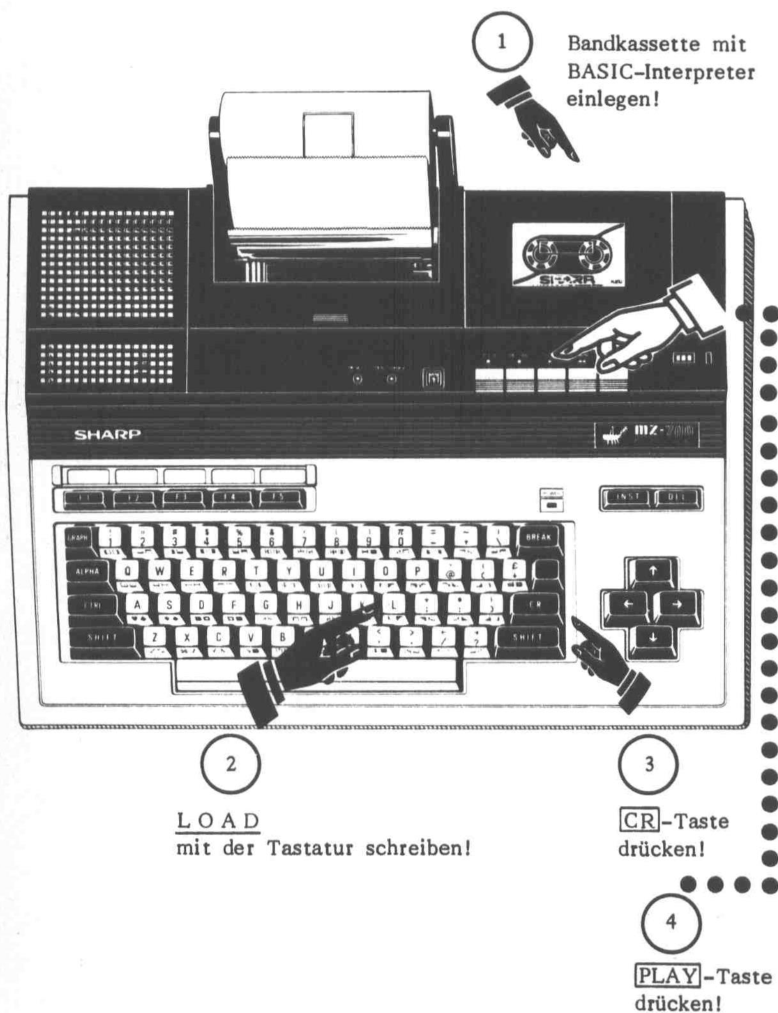
(g) Der Ladevorgang ist nach ca. 3,5 Minuten beendet, das Bandkassettenlaufwerk kommt zum Stehen, und der Bildschirm enthält folgenden Text:

---

1) von engl. "to load" - Aussprache: lood - Übersetzung: laden

2) CR = Abkürzung für engl. "Carriage Return" - Übersetzung: Wagenrücklauf (Die Bezeichnung für diese Taste wurde im Hinblick auf schreibmaschinenorientierte Peripheriegeräte zu einer Zeit eingeführt, als Bildschirmgeräte noch nicht verbreitet waren.)

Laden des BASIC-Interpreters



```
** MONITOR 1Z-013A **
*LOAD
⬇ PLAY
LOADING S-BASIC

   BASIC INTERPRETER  1Z-013B V1.0A

  COPYRIGHT (C) 1983 BY SHARP CORP.

 36439 BYTES FREE

Ready
```

(h) Wir betätigen jetzt am Bandgerät die STOP/EJECT-Taste, wodurch die PLAY-Taste zurückspringt, drücken dann auf die REWIND-Taste und erreichen damit, daß das Band zurückgespult wird. Eine nochmalige Betätigung der STOP/EJECT-Taste stellt das Gerät ab; wenn wir jetzt wiederum die STOP/EJECT-Taste betätigen, öffnen wir damit den Deckel des Bandkassettenlaufwerks und können die Kassette entnehmen, die wir geschützt aufbewahren sollten.

Wir arbeiten jetzt in erster Linie unter der "Leitung und Fürsorge" des BASIC-Interpreters und wollen nunmehr das folgende BASIC-Programm eingeben, das die programmtechnische Lösung des "ALARICH-Falls" gem. unserem Struktogramm darstellt, das wir für den direkten Vergleich auf der folgenden Seite 4-62 nochmals wiedergeben.

Bitte, vergessen Sie nicht, jede Programmzeile mit einer Betätigung der CR-Taste abzuschließen. Der Cursor springt dann automatisch immer in die nächste Zeile des Bildschirmbilds.

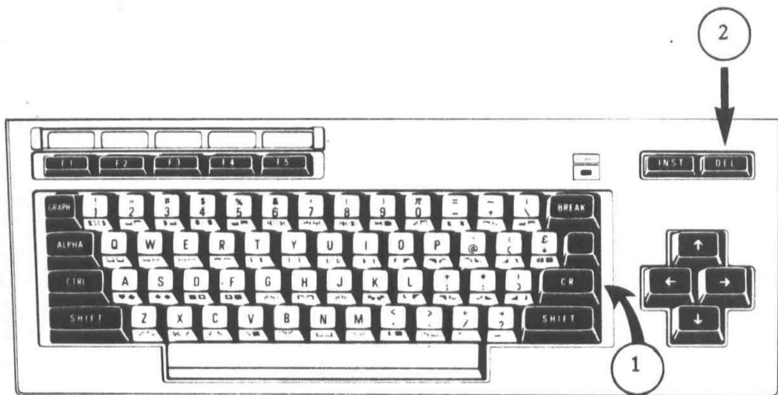
```
                        A = A + 1
                        B = A
                        AUSGEBEN A
                        B = B - 8
                        T = B / 5
                    T = GANZZAHLIG
                    B = B - T
                    B = B - 9
                    T = B / 5
                T = GANZZAHLIG
                B = B - T
                B = B - 10
                T = B / 5
            T = GANZZAHLIG
            B = B - T
            B = B - 11
            T = B / 5
        T = GANZZAHLIG
        B = B - T
        B = B - 12
        T = B / 5
    T = GANZZAHLIG
    B = B - T
B / 5 = GANZZAHLIG
AUSGEBEN "ERGEBNIS:"; A
```

Programmliste ("ALARICH"):

```
10 REM ***********
20 REM * ALARICH *
30 REM ***********
40 REM 1. Z-SCHLEIFE
50 REM ------------
60 LET A=A+1
70 LET B=A
80 PRINT A;
90 LET B=B-8
100 LET T=B/5
110 IF T=INT(T) THEN 130
120 GOTO 40
130 REM 2. Z-SCHLEIFE
140 REM ------------
150 LET B=B-T
160 LET B=B-9
170 LET T=B/5
180 IF T=INT(T) THEN 200
190 GOTO 40
200 REM 3. Z-SCHLEIFE
210 REM ------------
220 LET B=B-T
230 LET B=B-10
240 LET T=B/5
250 IF T=INT(T) THEN 270
260 GOTO 40
270 REM 4. Z-SCHLEIFE
280 REM ------------
290 LET B=B-T
300 LET B=B-11
310 LET T=B/5
320 IF T=INT(T) THEN 340
330 GOTO 40
340 REM 5. Z-SCHLEIFE
350 REM ------------
360 LET B=B-T
370 LET B=B-12
380 LET T=B/5
390 IF T=INT(T) THEN 410
400 GOTO 40
410 REM 6. Z-SCHLEIFE
420 REM ------------
430 LET B=B-T
440 IF B/5=INT(B/5) THEN 460
450 GOTO 40
460 REM ERGEBNISAUSGABE
470 REM --------------
480 PRINT "   ERGEBNIS:";A
490 END
```

Jede Programmzeile mit einer Betätigung der [CR]-Taste abschließen!

Tastatur



Erläuterungen: Für die ersten elementaren Programme interessieren uns nur jene Zeichen, die auf den Berührungsflächen der Tasten stehen, wie z. B. die Buchstaben. Sie werden nach Betätigung der jeweiligen Tasten auf dem Bildschirm wiedergegeben, wenn vorher keine besondere Funktionstaste betätigt wurde. Drückt man die SHIFT-Taste (sie ist wie bei der Schreibmaschine zweimal vorhanden) zusammen mit einer Buchstabentaste, so erhält man einen Kleinbuchstaben. Damit funktioniert die Tastatur genau umgekehrt wie jene der Schreibmaschine, was jedoch sinnvoll ist, weil man überwiegend die Großbuchstaben benötigt.

Enthält eine Taste auf der Berührungsfläche zwei Zeichen - wie z. B.  - dann erhält man das untere Zeichen (bei den beiden Beispieltasten also 8 oder 9) durch das Drücken nur dieser Taste und das obere Zeichen (bei den beiden Beispieltasten also (oder) - Zeichen, die wir auch in unserem "ALARICH"-Programm mehrmals benötigen!) durch gleichzeitiges Drücken einer SHIFT-Taste zusammen mit der ausgewählten Zeichentaste.

(4) Eingeben der Programmzeilen:

Nachdem Sie einen Befehl mit der Tastatur geschrieben haben, wird er durch das Drücken der CR-Taste in den Hauptspeicher übertragen. Der Cursor erscheint dann zu Anfang der folgenden beschreibbaren Zeile. (1)

(5) Korrektur von fehlerhaften Programmzeilen:

Wenn Sie vor Betätigung der CR-Taste feststellen, daß Sie sich verschrieben haben, dann können Sie durch Betätigung der DEL-Taste mit dem Cursor innerhalb der Zeile zurückgehen und dabei automatisch Zeichen löschen. (1) (2)

Wenn Sie die Zeile durch Betätigung der CR-Taste schon abgeschlossen haben, dann besteht die einfachste Art einer Korrektur darin, daß Sie (an beliebiger Stelle) die Zeile neu schreiben und die CR-Taste drücken. Der fehlerhafte Befehl im Hauptspeicher wird dann durch den korrekten Befehl überschrieben. (1) (1)

Beispiel:

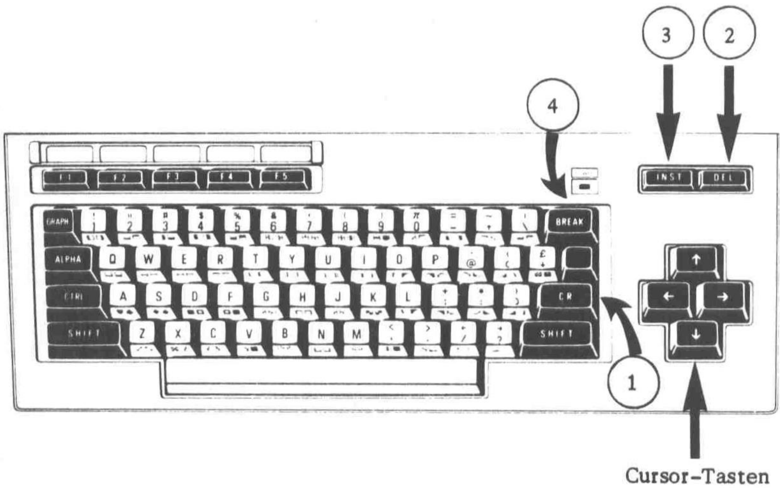
```
10 REM QUADRATZAHLEN
20 LETTT A$="QADD:+"
30 LET B$="="
40 LET A=A+1

20 LET A$="QUADR. V."
 :
```

An dieser Stelle entdecken Sie die Fehler in Zeile 20.

Sie korrigieren den Befehl einfach dadurch, daß Sie die Zeile neu schreiben und die CR-Taste drücken. Zwar ist dann auf dem Bildschirm die fehlerhafte Zeile noch zu sehen, im Arbeitsspeicher wurde sie jedoch durch den richtigen Befehl überschrieben. (Wenn Sie jetzt das Systemkommando LIST - ohne Zeilennummer - erteilen, können Sie sich davon überzeugen, daß im Arbeitsspeicher die "Ordnung" wieder hergestellt ist!) (1)

Tastatur


3
2
4
1
INST
DEL
GRAPH
ALPHA
CTRL
SHIFT
BREAK
CR
SHIFT
Cursor-Tasten

Soll aber eine Befehlszeile im Hauptspeicher gelöscht werden, so schreibt man nur die entsprechende Zeilennummer ohne einen Zusatz und drückt dann die [CR]-Taste. (1)

Eine weitere Korrektur-Möglichkeit besteht darin, daß man zunächst den Cursor mit Hilfe der Cursor-Tasten auf dem Bildschirm "in" den zu korrigierenden Befehl bewegt. Die Wirkungsweise der Cursor-Tasten ergibt sich aus den aufgedruckten Pfeilen.

Wenn man sich mit dem Cursor an der zu korrigierenden Stelle eines Befehls befindet, dann ist es möglich, mit der [DEL]-Taste das jeweilige Zeichen, (2)
das sich unmittelbar links vom Cursor befindet, zu löschen. Mit der [INST]-Taste läßt sich zusätzlicher (3)
Zwischenraum für einzufügende Zeichen erzeugen.

Nach erfolgter Korrektur wird der Befehl durch Drücken der [CR]-Taste in den Arbeitsspeicher (1)
zurückgeschrieben.

(6) Start des Programms:

Sie starten das korrekte Programm mit dem Systemkommando RUN. Der Programmlauf läßt sich

- unterbrechen durch das Niedergedrückthalten der [BREAK]-Taste, (4)
- vorzeitig beenden durch die gleichzeitige Betätigung der Tasten [SHIFT] und [BREAK], (5) (4)
- fortsetzen durch Erteilung des Systemkommandos CONT.

Als Verarbeitungsergebnis unseres "ALARICH"-Programms erhalten wir nach ca. 5 Minuten die gesuchte Zahl:

```
    ... 9362 9363 ERGEBNIS:
 9363
Ready
```

7. BASIC-Befehle und ihre Numerierung

LE 7.1

Wir wollen nun das **ÜBUNG**sprogramm zu Lektion 6 wieder aufgreifen, um anhand dieses Beispiels wichtige BASIC-Sprachelemente zu erläutern.

Unser Programm (vgl. folgende Seite!) besteht aus 49 Befehlen, die jeweils mit einer Zeilennummer beginnen. Die Zeilennumerierung ist zwingend vorgeschrieben. Durch sie wird

- die Reihenfolge festgelegt, in der die Befehle im Arbeitsspeicher stehen

  und

- die Reihenfolge vorgegeben, in der die Befehle ausgeführt werden sollen. (Abweichungen von dieser Reihenfolge sind natürlich möglich, z. B. durch Sprungbefehle.)

Programmliste ("ALARICH"):




```
10 REM ***********
20 REM * ALARICH *
30 REM ***********
40 REM 1. Z-SCHLEIFE
50 REM -------------
60 LET A=A+1
70 LET B=A
80 PRINT A;
90 LET B=B-8
100 LET T=B/5
110 IF T=INT(T) THEN 130
120 GOTO 40
130 REM 2. Z-SCHLEIFE
140 REM -------------
150 LET B=B-T
160 LET B=B-9
170 LET T=B/5
180 IF T=INT(T) THEN 200
190 GOTO 40
200 REM 3. Z-SCHLEIFE
210 REM -------------
220 LET B=B-T
230 LET B=B-10
240 LET T=B/5
250 IF T=INT(T) THEN 270
260 GOTO 40
270 REM 4. Z-SCHLEIFE
280 REM -------------
290 LET B=B-T
300 LET B=B-11
310 LET T=B/5
320 IF T=INT(T) THEN 340
330 GOTO 40
340 REM 5. Z-SCHLEIFE
350 REM -------------
360 LET B=B-T
370 LET B=B-12
380 LET T=B/5
390 IF T=INT(T) THEN 410
400 GOTO 40
410 REM 6. Z-SCHLEIFE
420 REM -------------
430 LET B=B-T
440 IF B/5=INT(B/5) THEN 460
450 GOTO 40
460 REM ERGEBNISAUSGABE
470 REM ---------------
480 PRINT "   ERGEBNIS:";A
490 END
```




Regeln für die Zeilennumerierung:

(1) Zulässige Zeilennummern liegen im Bereich von 1 bis 65 535.

(2) Es ist zweckmäßig, die Zeilennummern - wie in unserem Beispiel - in Zehnerschritten zu vergeben. Man hat dann "freie Nummern", um evtl. nachträglich zusätzliche Befehle einzuschieben.

(3) Unter einer Zeilennummer dürfen insgesamt nicht mehr als 255 Zeichen geschrieben werden. Aus Gründen der Übersichtlichkeit sollte man jedoch möglichst darauf verzichten, die Kapazität einer Bildschirmzeile zu überschreiten.

(4) Die Befehlszeile wird durch Drücken der CR-Taste abgeschlossen und im Arbeitsspeicher abgelegt. Der Cursor (= Blinkzeichen) erscheint dann am Anfang der folgenden beschreibbaren Zeile.

(5) Es ist zwar zulässig, unter einer Zeilennummer mehrere Befehle aufzuführen, die dann durch Doppelpunkte zu trennen sind.

Beispiel:

```
10 REM BEISPIEL:LET A$="QUADR. V."
20 LET B$=" IST":LET A=10
30 LET A=A+1:LET B=A*A:PRINT A$;A;B$;B
40 IF A<20 THEN 30
50 PRINT "ENDE DER BERECHNUNG":END
```

Aus Gründen der Übersichtlichkeit sollte man aber auf diese Schreibweise (mehrere Befehle pro Zeile) im allgemeinen verzichten.

(6) Anders als bei vielen BASIC-Rechnern, braucht man beim SHARP MZ-700 die einzelnen Befehlselemente nicht durch Leerstellen zu trennen.

Beispiel:

```
10REMBEISPIEL
20LETA$="QUADR. V."
30LETB$=" IST"
40LETA=10
50LETA=A+1
60LETB=A*A
70PRINTA$;A;B$;B
80IFA<20THEN50
90PRINT"ENDE DER BERECHNUNG"
100END
```

Wir empfehlen aber, zwischen den Befehlselementen immer dann eine Leerstelle zu ergänzen, wenn dadurch die Übersichtlichkeit und Lesbarkeit wesentlich verbessert wird.

Beispiel:

```
10 REM BEISPIEL
20 LET A$="QUADR. V."
30 LET B$=" IST"
40 LET A=10
50 LET A=A+1
60 LET B=A*A
70 PRINT A$;A;B$;B
80 IF A<20 THEN 50
90 PRINT "ENDE DER BERECHNUNG"
100 END
```

Manche Programmierer ergänzen zwischen den Befehlselementen sogar grundsätzlich eine Leerstelle.

Beispiel:

```
10 REM BEISPIEL
20 LET A$ = "QUADR. V."
30 LET B$ = " IST"
40 LET A = 10
```

```
50 LET A = A + 1
60 LET B = A * A
70 PRINT A$; A; B$; B
80 IF A < 20 THEN 50
90 PRINT "ENDE DER BERECHNUNG"
100 END
```

MERKE:

(1) BASIC-Befehle sind mit Zeilennummern im Bereich von 1 bis 65 535 zu versehen.

(2) Die Zeilennummern sollten möglichst in Zehnerschritten vergeben werden (10, 20, 30, 40, . . .) .

(3) Beim Schreiben der Befehle sollte man sich um Übersichtlichkeit bemühen und nicht unbedingt alle Möglichkeiten ausnutzen, die ein Rechner "erlaubt".

PE 7.1

Welche der folgenden Behauptungen ist/sind richtig?

A BASIC-Befehle sind mit Zeilennummern zu versehen, damit man ohne Schwierigkeiten überprüfen kann, wieviel Befehle pro Stunde geschrieben wurden. Wenn eine Leistungskontrolle nicht stattfinden soll, dann kann die Zeilennumerierung entfallen.

B Durch die zwingend vorgeschriebene Zeilennumerierung wird die Reihenfolge festgelegt, in der die Befehle im Arbeitsspeicher stehen und in der die Befehle ausgeführt werden sollen (Ausnahme: Sprungbefehle etc.).

C Zulässige Zeilennummern liegen im Bereich von 1 bis 65 535.

D Es ist zweckmäßig, Zeilennummern in Zehnerschritten (10, 20, 30, 40, . . .) zu vergeben, weil man so "freie Nummern" hat, um evtl. nachträglich zusätzliche Befehle einzuschieben.

E Die Zeilennumerierung muß, beginnend mit 0, aufsteigend in Einerschritten (0, 1, 2, 3, 4, . . .) erfolgen. Nur so beweist ein Programmierer seine Fähigkeit zum systematischen Denken.

F Es ist im allgemeinen empfehlenswert, pro Zeilennummer nur einen Befehl zu schreiben und die Befehlselemente immer dann durch Leerstellen voneinander abzuheben, wenn dadurch die Übersichtlichkeit und Lesbarkeit des Programms wesentlich verbessert wird.

G Falls man sich (z. B. aus Gründen der Arbeitsspeicherplatzersparnis) im Ausnahmefall doch einmal entschließen muß, unter einer Zeilennummer mehrere Befehle aufzuführen, so sind diese durch Doppelpunkte voneinander zu trennen.

Der/die Lösungsbuchstabe/n lautet/lauten

B, G, D, F, G . . . .

LÖS 7.1

Die Lösungsbuchstaben lauten

B, C, D, F, G .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 7.1** !

LE 7.2

Wir wissen nunmehr, daß Programmzeilen in aufsteigenden Zehnerschritten zu numerieren sind.

Aus welchen Elementen besteht der eigentliche Befehl?

Verallgemeinernd kann gesagt werden, daß in einem Befehl zwei Dinge anzugeben sind:

(1) W A S der Rechner machen soll
(= Operationsteil des Befehls),
z. B. lesen, ausgeben, vergleichen,

(2) W O M I T der Rechner etwas machen soll
(= Operandenteil des Befehls).

Erläuterungen:

Zu (1): Mit P R I N T (deutsch: drucke, gib aus) legt man beispielsweise fest, W A S geschehen soll.

Zu (2): Der Rechner soll z. B. mit dem Inhalt eines Speicherplatzes etwas machen, der den Namen A $ hat. Die Anweisung .. P R I N T A $ würde bedeuten, daß mit dem Inhalt des Speicherfeldes A $ (= W O M I T) etwas geschehen soll.

PE 7.2

Im Rahmen eines Programms wurde der folgende Befehl geschrieben:

```
.
.
110 PRINT A$
.
.
```

(1) Der obige Befehl erscheint unter der Zeilennummer

. . . .

(2) Der Operationsteil des Befehls lautet

. . . .

(3) Der Operationsteil bedeutet, daß

. . . . werden soll.

(4) Der Operandenteil des Befehls lautet

. . . .

(5) Der Operandenteil ist hier der Name eines Speicherplatzes. Nicht der Name des Speicherplatzes ist auszugeben, sondern

. . . .

LÖS 7.2

Im Rahmen eines Programms wurde der folgende Befehl geschrieben:

```
.
.
110 PRINT A$
.
.
```

(1) Der obige Befehl erscheint unter der Zeilennummer
110 .

(2) Der Operationsteil des Befehls lautet
PRINT .

(3) Der Operationsteil bedeutet, daß
ausgegeben werden soll.

(4) Der Operandenteil des Befehls lautet
A$.

(5) Der Operandenteil ist hier der Name eines Speicherplatzes. Nicht der Name des Speicherplatzes ist auszugeben, sondern
dessen Inhalt .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 7.2** !

ÜBUNGsprogramm zu Lektion 7

P r o b l e m s t e l l u n g:

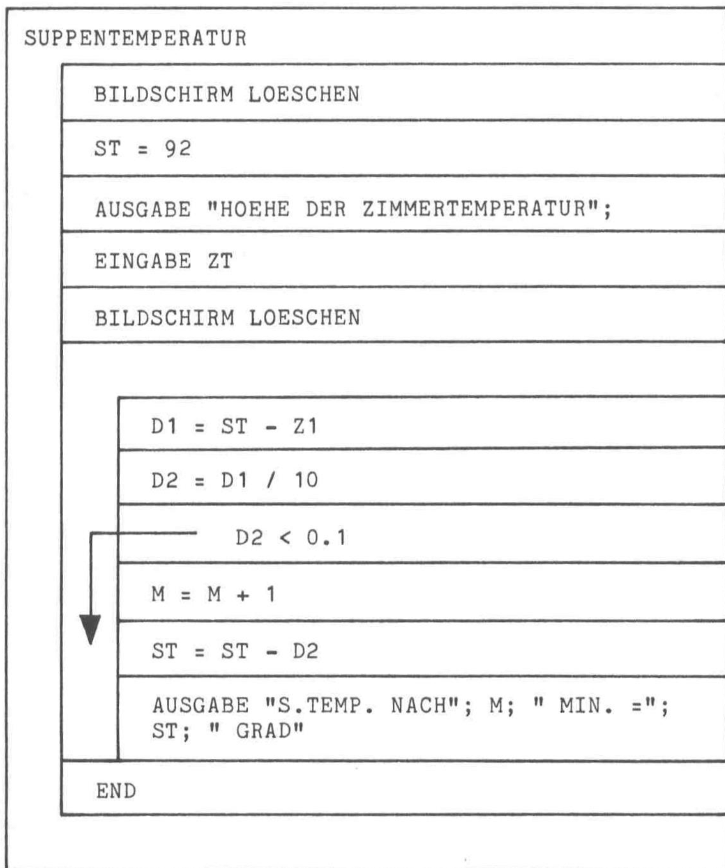
Eine Suppe, die unmittelbar nach dem Kochen (= 100° C) auf den Tisch gebracht wird, hat dann (durch Umfüllen etc.) eine Temperatur von 92° C. In jeder Minute verringert sich die Suppentemperatur um ca. 10 % der Differenz zur Zimmertemperatur. - Bei einer Zimmertemperatur von 20° C berechnet man also die Temperatur der Suppe nach 1 Minute Wartezeit folgendermaßen:

(1)	Ausgangstemperatur der Suppe:	92°
(2)	Zimmertemperatur:	20°
(3)	Differenz zwischen Ausgangstemperatur der Suppe und Zimmertemperatur (92° - 20°):	72°
(4)	10 % der Differenz:	7,2°
(5)	Suppentemperatur nach 1 Minute (92° - 7,2°) :	84,8°

P r o g r a m m b e s c h r e i b u n g (Abnahme der Suppentemperatur):

Das folgende Programm berechnet nach der Eingabe der Zimmertemperatur durch den Anwender die Suppentemperaturen in Minutenabständen, und zwar so lange, wie die Temperaturabnahme mindestens 0,1° pro Minute beträgt. Hierdurch gewinnt man einen genauen Überblick darüber, wieviel Zeit man verstreichen lassen kann, bevor man sich nach dem "Ruf der Hausfrau" zu Tisch begibt, um dann eine Suppentemperatur vorzufinden, die den eigenen kulinarischen Präferenzen entspricht.

Struktogramm:



Programmliste:

```
10 REM ********************
20 REM * SUPPENTEMPERATUR *
30 REM ********************
40 CLS
50 LET ST = 92
60 PRINT "HOEHE DER ZIMMERTEMPERATUR";
70 INPUT ZT
80 CLS
90 REM BEGINN DER M-SCHLEIFE
100 REM --------------------
110 LET D1 = ST - ZT
120 LET D2 = D1 / 10
130 IF D2 < 0.1 THEN 180
140 LET M = M + 1
150 LET ST = ST - D2
160 PRINT "S.TEMP. NACH"; M; " MIN. =";
ST; " GRAD"
170 GOTO 90
180 REM ENDE DER M-SCHLEIFE
190 REM ------------------
200 END
```

Erläuterungen:

Zeilen 10, 20, 30: Diese Befehle verursachen keine eigentlichen Verarbeitungsschritte, sondern dienen nur der Kennzeichnung und ggf. Erläuterung des Programms (vgl. hierzu Lektion 9).

Zeile 40: Hiermit löscht man den Bildschirm (CLS = Abkürzug für "clear screen"; von engl. "to clear" = reinigen, freimachen und von engl. "screen" = Bildschirm).

Zeile 50: Der numerischen Variablen ST (SuppenTemperatur) wird die Ausgangstemperatur der Suppe zugewiesen.

Zeile 60: Auf dem Bildschirm wird der Text

```
HOEHE DER ZIMMERTEMPERATUR
```

ausgegeben (vgl. hierzu Lektion 12), damit der Anwender weiß, was er

eingeben soll, wenn aufgrund des Befehls in

Zeile 70 der Rechner ein Fragezeichen druckt und so lange wartet, bis eine Eingabe getätigt wurde, die mit dem Drücken der [CR]-Taste abzuschließen ist (vgl. hierzu Lektion 15).

Das System legt den eingegebenen Wert in der nach dem Befehlswort I N P U T angegebenen Variablen ZT (= Zimmer Temperatur) ab.

Zeile 80: Löschung des Bildschirms

Zeilen 90, 100: Befehle zur Erläuterung der Programmstruktur

Zeile 110: Ermittlung der Differenz zwischen der aktuellen Suppen-Temperatur und der Zimmer Temperatur und Speicherung des Ergebnisses in der Variablen D1 (vgl. hierzu Lektion 10 und 11)

Zeile 120: Ermittlung eines Zehntels (10 %) von der in Zeile 110 ermittelten Temperaturdifferenz (vgl. Lektion 10 und 11)

Zeile 130: Falls der in Zeile 120 für D2 errechnete Wert kleiner ist als 0,1 , dann springen wir zum Ende der Schleife nach Zeile 180 - andernfalls wird automatisch der nächste Befehl bearbeitet (vgl. hierzu Lektion 18).

Zeile 140: Erhöhung der numerischen Variablen M um 1
(Da der Anfangswert von M = 0 ist, hat M nach der erstmaligen Bearbeitung dieses Befehls den Wert 1.)

Zeile 150: ST wird um D2 verringert.

Zeile 160: Dieser Befehl erzeugt auf dem Bildschirm die folgende Ausgabe (anläßlich der erstmaligen Abarbeitung der Anweisung), wenn in Zeile 70 beispielsweise 21 als Zimmertemperatur eingegeben worden ist:

```
S.TEMP. NACH 1 MIN. = 84.9 GRAD
```

Zeile 170: Sprung zur Zeile 90 (vgl. hierzu Lektion 18)

Zeilen 180, 190: Befehle zur Erläuterung der Programmstruktur

Zeile 200: Beendigung des Programms (vgl. hierzu Lektion 19)

Modifizierte Lösung:

Für eine Ausgabe der Ergebnisse auf dem Plotter-Drucker müssen wir lediglich das Befehlswort in Zeile 160 ändern, und zwar schreiben wir statt PRINT dann PRINT/P (vgl. hierzu Lektion 12, vor allem Seite 12-11).

Ergebnisse eines Programmlaufs (der modifizierten Lösung):

Bildschirm:

```
HOEHE DER ZIMMERTEMPERATUR? 21
```

(Eingabe des Benutzers)

Plotter-Drucker:

(vgl. folgende Seite)

Plotter-Drucker:

```
S.TEMP. NACH 1 MIN. = 84.9 GRAD
S.TEMP. NACH 2 MIN. = 78.51 GRAD
S.TEMP. NACH 3 MIN. = 72.759 GRAD
S.TEMP. NACH 4 MIN. = 67.5831 GRAD
S.TEMP. NACH 5 MIN. = 62.92479 GRAD
S.TEMP. NACH 6 MIN. = 58.732311 GRAD
S.TEMP. NACH 7 MIN. = 54.95908 GRAD
S.TEMP. NACH 8 MIN. = 51.563172 GRAD
S.TEMP. NACH 9 MIN. = 48.506855 GRAD
S.TEMP. NACH 10 MIN. = 45.756169 GRAD
S.TEMP. NACH 11 MIN. = 43.280552 GRAD
S.TEMP. NACH 12 MIN. = 41.052497 GRAD
S.TEMP. NACH 13 MIN. = 39.047247 GRAD
S.TEMP. NACH 14 MIN. = 37.242523 GRAD
S.TEMP. NACH 15 MIN. = 35.61827 GRAD
S.TEMP. NACH 16 MIN. = 34.156443 GRAD
S.TEMP. NACH 17 MIN. = 32.840799 GRAD
S.TEMP. NACH 18 MIN. = 31.656719 GRAD
S.TEMP. NACH 19 MIN. = 30.591047 GRAD
S.TEMP. NACH 20 MIN. = 29.631942 GRAD
S.TEMP. NACH 21 MIN. = 28.768748 GRAD
S.TEMP. NACH 22 MIN. = 27.991873 GRAD
S.TEMP. NACH 23 MIN. = 27.292686 GRAD
S.TEMP. NACH 24 MIN. = 26.663417 GRAD
S.TEMP. NACH 25 MIN. = 26.097076 GRAD
S.TEMP. NACH 26 MIN. = 25.587368 GRAD
S.TEMP. NACH 27 MIN. = 25.128631 GRAD
S.TEMP. NACH 28 MIN. = 24.715768 GRAD
S.TEMP. NACH 29 MIN. = 24.344191 GRAD
S.TEMP. NACH 30 MIN. = 24.009772 GRAD
S.TEMP. NACH 31 MIN. = 23.708795 GRAD
S.TEMP. NACH 32 MIN. = 23.437916 GRAD
S.TEMP. NACH 33 MIN. = 23.194124 GRAD
S.TEMP. NACH 34 MIN. = 22.974712 GRAD
S.TEMP. NACH 35 MIN. = 22.77724 GRAD
S.TEMP. NACH 36 MIN. = 22.599516 GRAD
S.TEMP. NACH 37 MIN. = 22.439565 GRAD
S.TEMP. NACH 38 MIN. = 22.295608 GRAD
S.TEMP. NACH 39 MIN. = 22.166047 GRAD
S.TEMP. NACH 40 MIN. = 22.049443 GRAD
S.TEMP. NACH 41 MIN. = 21.944498 GRAD
```

8. Variablen und Konstanten

LE 8.1

Im Operandenteil eines Befehls können Konstanten und/oder Variablen aufgeführt sein. Wir wollen uns zunächst den Konstanten zuwenden.

Konstanten sind jene Daten eines BASIC-Programms, die sich nicht verändern. Grundsätzlich unterscheiden wir zwei Konstantentypen:

(1) numerische Konstanten und

(2) Zeichenkettenkonstanten (sog. "Strings").

Zu (1) numerische Konstanten:

Numerische Konstanten können wiederum in drei Formaten vorkommen:

	Beispiele:
- als Ganzzahl	218 -163
- als Festkommazahl [1]	218.75 -7.6
- als Gleitkommazahl [1]	1.75E+09 1.46E-09

1) Achtung: Abweichend von den deutschen Gewohnheiten wird in BASIC - wie in den USA üblich - die Dezimalstelle durch einen Punkt und nicht durch ein Komma gekennzeichnet. Deshalb findet man in der Literatur zu BASIC bisweilen auch die Bezeichnungen Festpunktzahl (statt Festkommazahl) und Gleitpunktzahl (statt Gleitkommazahl).

Erläuterungen: Die Gleitkommazahl 1.75E+08 steht für 1.75 x 10^8, das bedeutet 1.75 x 100 000 000, das entspricht 175 000 000.

Die Gleitkommazahl 1.46E-09 steht für 1.46 x 10^{-9}, das bedeutet 1.46 x 0.000 000 001, das entspricht 0.000 000 001 46.

Zu (2) Zeichenkettenkonstanten:

Eine Zeichenkettenkonstante (= String) ist eine Folge von beliebigen Zeichen (maximal 255), die in Anführungsstrichen eingeschlossen sind. Die Anführungszeichen kennzeichnen Anfang und Ende der Zeichenkettenkonstanten, ohne selbst Bestandteil der Konstanten zu sein.

Beispiele:

```
"FONTANE: EFFI BRIEST - EINE INTERPRETATION VON"
"ELSBETH HAMANN (IM VERLAG R. OLDENBOURG, MUENCHEN)"
"ISBN: 3-486-19961-7"
"GESAMTSUMME"
"MEINE TOCHTER HEISST MAJA AHN CHA RIM HAMANN"
"BITTE 'W'-TASTE DRUECKEN!"
"LOGIK DER PROGRAMMIERUNG"
"EIN BUCH VON G. O. HAMANN"
"45,-- DM"
```

Achtung: Auch wenn eine Zeichenkettenkonstante nur aus Ziffern besteht, kann sie gleichwohl nicht für Rechenoperationen verwendet werden. Rechenoperationen kann man nur mit numerischen Konstanten ausführen. (Allerdings ist es möglich, mit Hilfe der vordefinierten Funktion VAL(X$) eine Umwandlung der führenden Ziffern eines Strings in einen numerischen Wert vorzunehmen; vgl. hierzu die Ausführungen in Lektion 24!)

Ergänzender Hinweis für den forgeschrittenen Leser:

Neben den beschriebenen Konstantentypen kennt der SHARP MZ-700 außerdem

(1) die Systemkonstante π ("SHIFT"-Taste + [π/0]-Taste drücken!), die den Wert 3.1415927 repräsentiert, und

(2) 4-ziffrige Hexadezimal-Konstanten, die durch ein zusätzlich vorangestelltes Dollarzeichen ($) zu kennzeichnen sind. Hexadezimal-Konstanten dürfen nur mit LIMIT, POKE, PEEK und USR verwendet werden. Außerdem sind Kenntnisse der Maschinensprache erforderlich. (Vgl. hierzu die Systemunterlagen des Herstellers!)

PE 8.1

(Bitte, die linke Seite abdecken!)

(1) Ein Befehl besteht generell aus einem

Operationsteil (WAS soll der Rechner machen?) und einem

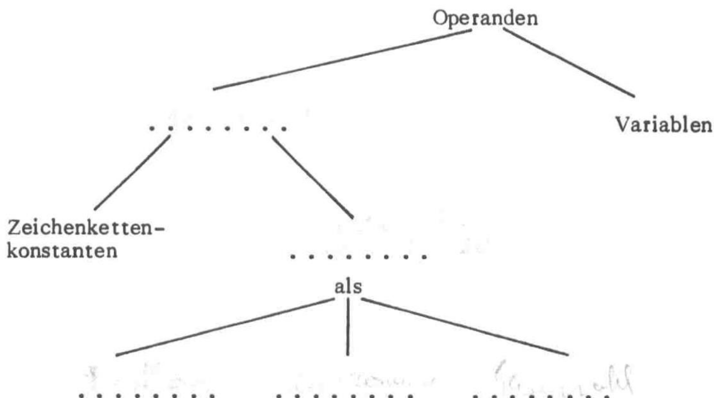
Operandenteil (WOMIT soll der Rechner etwas machen?).

Im Operandenteil eines Befehls können

. . . . und/oder . . . .

aufgeführt sein.

(2) Vervollständigen Sie die folgende Systematik!



(3) Kennzeichnen Sie im Programm auf der folgenden Seite jene Befehle, die eine Konstante enthalten.

Bestimmen Sie dann den Konstantentyp (numerische Konstante oder Zeichenkettenkonstante), und spezifizieren Sie bei numerischen Konstanten das Format (Ganzzahl oder Festkommazahl oder Gleitkommazahl)!

BASIC-Programm:	Lösung:
`10 REM ******************`	....
`20 REM * QUADRATZAHLEN  *`	....
`30 REM * FUER DIE ZAHLEN*`	....
`40 REM * VON 10 BIS 100 *`	....
`50 REM ******************`	....
`60 CLS`	....
`70 LET A$ = "QUADRAT VON"`	....
`80 LET B$ = " ="`	....
`90 LET A = 10`	....
`100 LET A = A + 1`	....
`110 LET B = A * A`	....
`120 PRINT A$; A; B$; B`	....
`130 IF A < 100 THEN 100`	....
`140 PRINT`	....
`150 PRINT "SCHLUSS"`	....
`160 END`	....

LÖS 8.1

(1) Ein Befehl besteht generell aus einem

Operationsteil (W A S soll der Rechner machen?) und einem

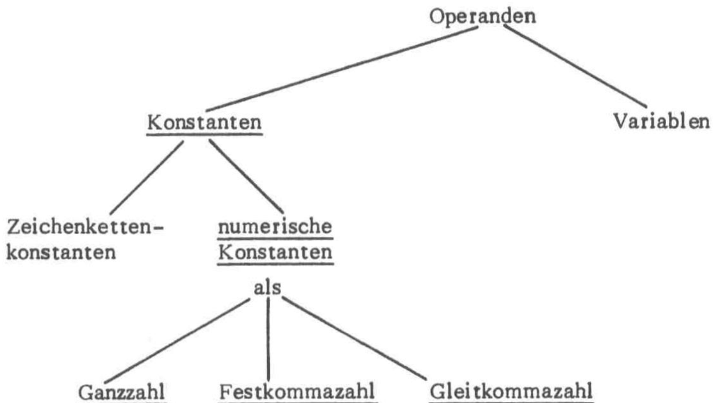
Operandenteil (W O M I T soll der Rechner etwas machen?).

Im Operandenteil eines Befehls können

K o n s t a n t e n und/oder V a r i a b l e n

aufgeführt sein.

(2) So sieht die vervollständigte Systematik aus:



(3) Bestimmung der Konstanten:

BASIC-Programm:

```
10 REM *****************
20 REM * QUDRATZAHLEN   *
30 REM * FUER DIE ZAHLEN*
40 REM * VON 10 BIS 100 *
50 REM *****************
60 CLS
70 LET A$ = "QUADRAT VON"
80 LET B$ = " ="
90 LET A = 10
100 LET A = A + 1
110 LET B = A * A
120 PRINT A$; A; B$; B
130 IF A < 100 THEN 100
140 PRINT
150 PRINT "SCHLUSS"
160 END
```

Erläuterungen:

- Zeilen 10–50: In den Zeilen 10 bis 50 handelt es sich nicht um Konstanten, sondern um Kommentare im Rahmen von REM-Anweisungen.
- Zeile 70, "QUADRAT VON": Zeichenkettenkonstante
- Zeile 80, " =": Zeichenkettenkonstante
- Zeile 90, 10: numerische Konstante als Ganzzahl
- Zeile 100, 1: numerische Konstante als Ganzzahl
- Zeile 130, THEN 100: Achtung: Dies ist keine Konstante im Sinne von BASIC, sondern eine Zeilennummer!
- Zeile 130, A < 100: numerische Konstante als Ganzzahl
- Zeile 150, "SCHLUSS": Zeichenkettenkonstante

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 8.1** !

LE 8.2

Wir haben gelernt, daß im Operandenteil eines Befehls Konstanten und/oder Variablen aufgeführt sein können. Nachdem wir die Konstanten behandelt haben, wollen wir uns den Variablen zuwenden.

In BASIC hat der Begriff "Variable" eine andere Bedeutung als in der Mathematik, und zwar sind Variablen lediglich Speicherfelder, deren Inhalte sich während eines Programmlaufs ändern können. Man unterscheidet

(1) numerische Variablen und

(2) String-Variablen (= Zeichenketten-Variablen).

Zu (1) numerische Variablen:

In unserem Programm in der vorangegangenen **PE** (vgl. auch die folgende Seite!) haben wir beispielsweise

- die numerische Variable A (anders ausgedrückt: ein Datenfeld mit dem Namen A) auf den Anfangswert 10 gesetzt (Zeile 90),

- dann haben wir diese Variable um 1 erhöht (Zeile 100),

- um anschließend A mit sich selbst zu multiplizieren (genauer: wir multiplizieren den Inhalt des Speicherfeldes mit dem Namen A mit sich selbst). Das Ergebnis dieser Multiplikation legen wir in der Variablen B ab (Zeile 110).

BASIC-Programm:

```
10 REM ******************
20 REM * QUADRATZAHLEN   *
30 REM * FUER DIE ZAHLEN*
40 REM * VON 10 BIS 100 *
50 REM ******************
60 CLS
70 LET A$ = "QUADRAT VON"
80 LET B$ = " ="
90 LET A = 10
100 LET A = A + 1
110 LET B = A * A
120 PRINT A$; A; B$; B
130 IF A  < 100 THEN 100
140 PRINT
150 PRINT "SCHLUSS"
160 END
```

Die Namen für numerische Variablen werden bei sehr vielen BASIC-Rechnersystemen - und so auch beim SHARP MZ-700 - folgendermaßen gebildet:

- ein beliebiger Buchstabe oder
- zwei beliebige Buchstaben oder
- ein Buchstabe mit einer angehängten Ziffer .

Beispiele:

richtig: A S Z AZ BB SE
XY PU RA A1 Z9 F7

falsch:
1A (zuerst einen Buchstaben verwenden!)
P33 (maximal 1 Buchstaben und 1 Ziffer verwenden!)
66 (zuerst einen Buchstaben verwenden!)
ABC (maximal 2 Buchstaben verwenden!)

Zu (2) String-Variablen:

Man bildet die Namen für String-Variablen wie jene für numerische Variablen - nur wird zusätzlich ein abschließendes $-Zeichen (gewissermaßen ein stilisiertes "S" für String) angehängt.

Beispiele:

richtig:	A$	S$	Z$	AZ$	BB$	FF$	HE$
	A1$	X1$	X2$	Y1$	F7$	F8$	HA$

falsch: F$G (Das $-Zeichen muß das letzte Zeichen sein!)

Auch Konstanten kann man Namen geben und sie dann wie Variablen verwenden, obwohl sie ihren Wert nicht ändern. In unserem Programm auf Seite 8-09 haben wir in Zeile 70 die Zeichenkettenkonstante "QUADRAT VON" der Variablen A$ zugewiesen. (Anders ausgedrückt: Wir haben den Text "QUADRAT VON" in einem Datenfeld mit dem Namen A$ abgelegt.) Und in Zeile 80 haben wir die Zeichenkettenkonstante " =" der Variablen B$ zugewiesen. (Anders ausgedrückt: Wir haben die Zeichen - nämlich 1 Leerstelle und 1 Gleichheitszeichen - in dem Feld mit dem Namen B$ abgelegt.)

Den Inhalt dieser Datenfelder geben wir dann - zusammen mit dem Inhalt der Variablen A und B - mit dem PRINT-Befehl in Zeile 120 aus. Die Konstante "SCHLUSS" hingegen haben wir keiner Variablen zugewiesen, sondern in Zeile 150 unmittelbar ausgegeben. Das Sprichwort "Viele Wege führen nach Rom" gilt also auch für die Programmiersprache BASIC!

In diesem Zusammenhang muß noch erwähnt werden, daß man Zeichenketten (Variablen und/oder Konstanten) mit Hilfe des Operators + verbinden kann. Wenn beispielsweise A$ = "BLOCK" und B$ = "SATZ", dann enthält nach der Operation C$ = A$ + B$ das Datenfeld C$ die Zeichenkettenkonstante "BLOCKSATZ": Ein so in einer Variablen zusammengesetzter String darf aus bis zu 255 Zeichen bestehen. (Man bezeichnet diese Verkettung bisweilen auch mit dem Fremdwort "Konkatenation"; von lat. "con" = zusammen und "catena" = Kette.)

Achtung: Anders als andere Programmiersprachen setzt BASIC die numerischen Variablen zu Beginn des Programmlaufs auf 0 (Null). Und wird der Inhalt einer String-Variablen ausgegeben, bevor ihr Zeichen zugewiesen worden sind, erscheinen nur Leerstellen.

Zusammenfassung

Variablen

(1) numerische Variablen			(2) String-Variablen		
Beispiele:	A	B	Beispiele:	A$	F$
	F1	Z3		A1$	Y3$
	GD	XY		TE$	KK$

Ergänzender Hinweis für den fortgeschrittenen Leser:

(Der Anfänger sollte die folgenden Ausführungen übergehen und statt dessen die **PE 8.2** auf Seite 8-13 bearbeiten!)

Wie wir auf Seite 8-02 ausgeführt haben, besteht ein String aus einer Folge von beliebigen Zeichen. Die Anführungszeichen kennzeichnen Anfang und Ende der Zeichenkettenkonstanten, ohne selbst Bestandteil der Konstanten zu sein. Daraus folgt indirekt schon, daß innerhalb des Strings Anführungszeichen nicht verwendet werden dürfen.

1. Beispiel: "SAVE "ALARICH" "

ist ein unzulässiger String, da sich innerhalb der begrenzenden Anführungszeichen weitere Anführungszeichen befinden!

In sehr vielen Fällen wird man sich mit Apostrophen behelfen können.

2. Beispiel: "SAVE 'ALARICH' "

wäre zwar eine zulässige Zeichenkette; allerdings ein unzulässiges Aussehen eines BASIC-Kommandos.

Will man aber beispielsweise im Rahmen eines Programms dem Anwender mitteilen, mit welchem Kommando er ein im Arbeitsspeicher befindliches Programm auf der Bandkassette abspeichern könnte, so würde die Ausgabe des Strings

"SAVE 'ALARICH' "

das Erscheinungsbild

SAVE 'ALARICH'

erzeugen - ein Kommando, das der Rechner zurückweist, denn nur

SAVE "ALARICH"

ist für das System verständlich.

In solchen Fällen kann man sich der Konkatenation bedienen und die Anführungsstriche im String mit der vordefinierten Funktion CHR$(X) erzeugen (vgl. hierzu Lektion 24!). Die vordefinierte Funktion CHR$(X) generiert aufgrund des in den Klammern angegebenen Dezimalwertes das gem. ASCII-Code zugeordnete Zeichen; in unserem Fall würde CHR$(34) Anführungsstriche hervorbringen.

Die Ausgabe der Zeichenkette

SAVE "ALARICH"

könnte beispielsweise mit den folgenden Befehlen erfolgen:

```
10 LET A$ = CHR$(34)
20 LET B$ = "SAVE "
30 LET D$ = "ALARICH"
40 LET E$ = B$ + A$ + D$ + A$
50 PRINT "DAS KOMMANDO LAUTET:"
60 PRINT E$
RUN
DAS KOMMANDO LAUTET:
SAVE "ALARICH"
Ready
```

PE 8.2

Entscheiden Sie, ob die angegebenen Namen für eine Variable und ggf. für welchen Variablentyp verwendet werden dürfen!

Machen Sie ein Kreuz in der jeweiligen Spalte!

Namen	Unzulässige Bezeichnung für Variable	Zulässige Bezeichnung für numerische Variable	Zulässige Bezeichnung für String-Variable
A			
B			
C			
AB			
1A			
FF			
$BB			
$B			
$B1			
BB$			×
B$			×
B1$			×
ZZ		×	
ZZ9	(×)		
ZZ9$	(×)		

LÖS 8.2

Namen	Unzulässige Bezeichnung für Variable	Zulässige Bezeichnung für numerische Variable	String-Variable
A		X	
B		X	
C		X	
AB		X	
1A	X		
FF		X	
$BB	X		
$B	X		
$B1	X		
BB$			X
B$			X
B1$			X
ZZ		X	
ZZ9	X [1)]		
ZZ9$	X [1)]		

1) Vgl. hierzu aber auch die Ausführungen auf Seite 8-15 f. !

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 8.2** !

LE 8.3

In der vorangegangenen **LE 8.2** haben wir gelernt, daß der Name für eine numerische Variable folgendermaßen gebildet wird:

- ein beliebiger Buchstabe oder
- zwei beliebige Buchstaben oder
- ein Buchstabe mit einer angehängten Ziffer .

String-Variablen unterscheiden sich von den numerischen Variablen durch das zusätzlich angefügte $-Zeichen.

Wir müssen die obigen Regeln nun um die folgenden Hinweise ergänzen.

(1) Variablennamen und reservierte BASIC-Wörter

Reservierte BASIC-Wörter - das sind Wörter, die in BASIC eine festgelegte Bedeutung haben - dürfen **nicht** als Variablennamen verwendet werden (z. B. IF oder ON).

(2) Variablennamen mit mehr als zwei Zeichen

Beim SHARP MZ-700 ist es auch zulässig, Variablennamen zu vergeben, die aus mehr als zwei Zeichen bestehen.

Beispiele:	ARMLEUCHTER	ARKUS	HAMANN
	ZAHL	TEXT$	GUENTER$

Dabei ist aber zu beachten, daß nur die ersten beiden Zeichen (und ggf. das angehängte Spezifikationszeichen $) signifikant sind. Das bedeutet, daß der Rechner für die "interne Verwaltung" die übrigen Zeichen nicht beachtet, und daher für ihn ARMLEUCHTER und ARKUS dieselbe Variable wäre (beide Variablennamen beginnen mit den zwei

signifikanten Zeichen AR !).

Außerdem ist es nicht zulässig, daß ein längerer Variablenname irgendein reserviertes BASIC-Wort enthält, weil der Interpreter dies als Aufforderung zu einer spezifischen Operation deutet. Beispielsweise wären die reservierten BASIC-Wörter ON und TO in dem Variablennamen KONTO enthalten, der daher (aus zwei Gründen!) unzulässig ist. Wenn man sich dahingehend einschränkt, nur die signifikanten Zeichen für Variablennamen zu vergeben, dann ist es vergleichsweise einfach, die sechs unzulässigen (weil reservierten) Zeichenkombinationen, nämlich FN, IF, LN, ON, TI$ (nicht: TI!) und TO, von der Verwendung auszuschließen.

Bei Benutzung längerer Variablennamen muß man hingegen wesentlich achtsamer sein und ständig überprüfen, ob darin eine Zeichenfolge enthalten ist, die einem der zahlreichen reservierten BASIC-Wörter entspricht. Allerdings ist der Interpreter bei der Aufdeckung fehlerhafter Variablennamen sehr behilflich und meldet ggf. "Syntax error in ..." (= Schreibfehler in Zeile ...).

(3) Ganzzahl-Variablen

Manche BASIC-Systeme kennen noch zusätzlich einen speziellen numerischen Variablentyp, nämlich Ganzzahl-Variablen, die durch das angehängte Zeichen % gekennzeichnet werden. Solche Variablen sind aber nur zulässig für Ganzzahlen im Bereich von -32 768 bis +32 767. Man verwendet sie vor allem aus Gründen der Arbeitsspeicherplatzersparnis. Das System SHARP MZ-700 kennt diesen Variablentyp nicht. Wir halten diese Vereinfachung für sehr sinnvoll, weil dem Anwender beim SHARP MZ-700 ein so großer freier Arbeitsspeicher zur Verfügung steht, daß Einsparungsmaßnahmen durch gelegentliche Verwendung von Ganzzahlvariablen nicht erforderlich sind.

(4) System-Variablen

System-Variablen sind reservierte BASIC-Wörter für Variablen,

- die der Rechner immer selbsttätig reserviert und
- deren Inhalt automatisch geändert wird.

Der SHARP MZ-700 kennt vier System-Variablen, nämlich ERN, ERL, SIZE und TI$. (ERN und ERL werden wir in Lektion 22 behandeln!)

Die 6-ziffrige Variable TI$ enthält den jeweiligen Wert der eingebauten Systemuhr (24-Stunden-Modus). Mit Hilfe des Befehls

```
TI$ = "160700"
```
[1]

können wir beispielsweise die Systemuhr auf 16 Uhr - 07 Minuten - 00 Sekunden einstellen, um anschließend die aktuelle Zeit zu beliebigen Terminen mit

```
PRINT TI$
```

abzurufen. [2]

Die Variable SIZE enthält die augenblicklich freie Arbeitsspeicherkapazität (in Bytes[3]), die sich mit jeder eingegebenen Programmzeile verringert. Durch

```
PRINT SIZE
```

können wir uns diese Information jederzeit ausgeben lassen.

---

1) Für die Wertzuweisung nach TI$ (vgl. hierzu auch Lektion 10) darf bei der uns vorliegenden BASIC-Version - abweichend von den Angaben an einigen Stellen der Systemunterlagen des Herstellers - das Befehlswort LET nicht verwendet werden; statt LET TI$ = "......" also nur TI$ = "......" schreiben! - Auch die direkte Eingabe nach TI$ mit Hilfe von INPUT (vgl. hierzu Lektion 15) ist in der von uns eingesetzten BASIC-Version nicht möglich. Statt dessen muß der "Umweg" über eine "Hilfsvariable" beschritten werden, also z. B. folgendermaßen:

```
.
.
70 INPUT H$
80 TI$ = H$
.
.
```

2) Wenn man einen Befehl ohne Zeilennummer eingibt, dann wird er zwar sofort ausgeführt, aber nicht gespeichert. Man bezeichnet diese Verwendung des Rechners als "direkte Betriebsart" bzw. "direkten Modus". Die Nutzung des Systems durch ein gespeichertes Programm nennt man hingegen den "indirekten Modus".

3) Ein Byte ist eine Speicherstelle, die sich aus 8 Bits zusammensetzt. Für das Programmieren mit BASIC sind genaue Kenntnisse hierüber nicht unbedingt erforderlich.

Zusammenfassung

Variablen

(1) numerische Variablen	(2) String-Variablen	(3) System-Variablen
Beispiele:	Beispiele:	ERN ERL
A B X	A$ B$ X$	TI$ SIZE
F1 Z3 GP	A1$ Y6$ DU$	

Ergänzender Hinweis für den fortgeschrittenen Leser:

(Der Anfänger sollte die folgenden Ausführungen übergehen!)

Wie für jeden Rechner, so gibt es auch für den SHARP MZ-700 Zahlenbereichsgrenzen, die nicht überschritten werden dürfen.

Höchste zulässige Gleitkommazahl: ±1.7014118E+38

Kleinste zulässige Gleitkommazahl: eine Mantisse mit dem Exponententeil E-38

Zahlen außerhalb dieses Bereichs verursachen die folgende Systemfehlermeldung:

Over flow error in ...

Für alle üblichen Anwendungsfälle ist das Überschreiten der gekennzeichneten Grenzen aber nicht zu erwarten.

PE 8.3

(1) Entscheiden Sie, ob die angegebenen Namen für eine Variable und ggf. für welchen Variablentyp verwendet werden dürfen!

Machen Sie ein Kreuz in der jeweiligen Spalte!

Name	Unzulässige Bezeichnung für Variable	Zulässige Bezeichnung für num. Variable	Zulässige Bezeichnung für String-Variable
AB		x	
AB$			x
A$B	x		
17	x		
18$	x		
DM		x	
DM$			x
$DM	x		
??	x		
??$	x		
$&$	x		
! FN	←	x	
IF	x		
! LN	←	x	
ON	x		
TO	x		

(2) Vervollständigen Sie den folgenden Befehl, mit dem die Größe der noch freien Arbeitsspeicherkapazität (in Bytes) ermittelt werden kann!

PRINT ....

(3) Vervollständigen Sie den folgenden Befehl, mit dem der Wert der eingebauten Systemuhr ausgegeben werden soll!

PRINT ....

LÖS 8.3

(1)

Name	Unzulässige Bezeichnung für Variable	Zulässige Bezeichnung für num. Variable	Zulässige Bezeichnung für String-Variable
AB		X	
AB$			X
A$B	X		
17	X		
18$	X		
DM		X	
DM$			X
$DM	X		
??	X		
??$	X		
$&$	X		
FN	X		
IF	X		
LN	X		
ON	X		
TO	X		

(2) Mit folgendem Befehl ermittelt man die Größe der noch freien Arbeitsspeicherkapazität (in Bytes):

PRINT SIZE .

(3) Mit dem folgenden Befehl kann man den Wert der eingebauten Systemuhr ausgeben lassen:

PRINT TI$.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 8.2** !

ÜBUNGsprogramm zu Lektion 8

Programmbeschreibung ("UHRZEIT1" und "UHRZEIT2"):

Das Programm "UHRZEIT1" gibt nach der Eingabe der aktuellen Uhrzeit in Sekundenabständen die jeweilige Uhrzeit aus, und zwar in der Form (Beispiel)

```
16 UHR 47 MINUTEN 18 SEKUNDEN
```

Programmliste:

(Die folgende Programmliste wurde aufgrund des Kommandos LIST/P auf dem Plotter-Drucker ausgegeben. - Das Kommando LIST/P erlaubt die Verwendung der gleichen Zusätze wie das Kommando LIST; vgl. hierzu die Ausführungen in Lektion 6, insbesondere Seite 6-05 und auf Seite Anhang-09 f.!)

```
10 REM ************
20 REM * UHRZEIT1 *
30 REM ************
40 CLS
50 REM EINGABE DER AKTUELLEN UHRZEIT
60 REM ----------------------------
70 PRINT : PRINT
80 PRINT "BITTE UHRZEIT EINGEBEN?"
90 PRINT
100 PRINT "FORM: HHMMSS"
110 PRINT
120 PRINT TAB(4);
130 INPUT UZ$
```

```
140 TI$ = UZ$
150 CLS
160 PRINT : PRINT
170 REM AUSGABE DER UHRZEIT
180 REM -------------------
190 PRINT TAB(5);
200 PRINT LEFT$(TI$, 2);
210 PRINT " UHR ";
220 PRINT MID$(TI$, 3, 2);
230 PRINT " MINUTEN ";
240 PRINT RIGHT$(TI$, 2);
250 PRINT " SEKUNDEN"
260 REM
270 REM WARTESCHLEIFE, BIS SEKUNDEN-
280 REM ZEIGEER 1 SEKUNDE VORGERUECKT
290 REM IST - DANN SPRUNG
300 REM NACH ZEILE 150
310 REM ----------------------------
320 LET AZ$ = RIGHT$(TI$, 2)
330 IF RIGHT$(TI$, 2) <> AZ$ THEN 150
340 GOTO 330
```

Erläuterungen:

(Nachdem das Programm mit RUN gestartet wurde, läuft es "endlos", es sei denn, man unterbricht die Ausführung durch gleichzeitige Betätigung der Tasten [SHIFT] und [BREAK].)

Zeilen 10, 20, 30: Diese Befehle verursachen keine eigentlichen Verarbeitungsschritte, sondern dienen nur der Kennzeichnung und ggf. Erläuterung des Programms (vgl. hierzu Lektion 9).

Zeile 40: Löschung des Bildschirms

Zeilen 50, 60: Befehle zur Erläuterung des Programms

Zeile 70: Ausgabe von zwei Leerzeilen

Zeile 80: Auf dem Bildschirm wird der Text

BITTE UHRZEIT EINGEBEN!

ausgegeben (vgl. hierzu Lektion 12), anschließend in

Zeile 90 eine Leerzeile und dann in

Zeile 100 der Text

FORM: HHMMSS ,

damit der Anwender weiß, was er eingeben soll, wenn aufgrund des Befehls in

Zeile 130 der Rechner ein Fragezeichen druckt und so lange wartet, bis eine Eingabe getätigt wurde, die mit dem Drücken der CR-Taste abzuschließen ist (vgl. hierzu Lektion 15).

Vor der Bearbeitung des INPUT-Befehls wurde in

Zeile 110 eine Leerzeile ausgegeben und durch den Befehl in

Zeile 120 erreichen wir einen Tabulator-Sprung (vgl. hierzu Lektion 24), so daß der Cursor unmittelbar unter dem Beginn des Strings "HHMMSS" positioniert ist:

```
BITTE UHRZEIT EINGEBEN!

FORM: HHMMSS

      ? ▒
```

Der String "HHMMSS" wurde in Anlehnung an die englische Sprache gebildet, weil die Zeichenkettenkonstante "SSMMSS" (d. h. 2 Ziffern für Stunden, 2 Ziffern für Minuten, 2 Ziffern für Sekunden) nicht eindeutig ist.

Zeile 140: Zuweisung der in Zeile 130 eingegebenen Uhrzeit zur System-Variablen TI$

Zeile 150: Löschung des Bildschirms

Zeile 160: Ausgabe von zwei Leerzeilen

Zeilen 170, 180: Befehle zur Erläuterung des Programms

Zeile 190: Tabulator-Sprung auf die 6. Schreibstelle, d. h. Ausgabe

von 5 Leerstellen (vgl. hierzu Lektion 24)

Zeile 200: Von der System-Variablen TI$ werden 2 Zeichen von links (= Stunden) abgetrennt und ausgegeben (vgl. hierzu Lektion 12 und Lektion 24).

Zeile 210: Ausgabe des Strings " UHR " (vgl. hierzu Lektion 12)

Zeile 220: Aus der Mitte der System-Variablen TI$ werden, beginnend mit dem 3. Zeichen, 2 Zeichen (= Minuten) abgetrennt und ausgegeben (vgl. hierzu Lektion 12 und Lektion 24).

Zeile 230: Ausgabe des Strings " MINUTEN " (vgl. hierzu Lektion 12)

Zeile 240: Von der System-Variablen TI$ werden 2 Zeichen von rechts (= Sekunden) abgetrennt und ausgegeben (vgl. hierzu Lektion 12 und Lektion 24).

Zeile 250: Ausgabe des Strings " SEKUNDEN"

Zeilen 260 - 310: Befehle zur Erläuterung des Programms

Zeile 320: Der String-Variablen AZ$ (Alte Zeit) werden von der System-Variablen TI$ 2 Zeichen von rechts (= Sekunden) zugewiesen (vgl. hierzu Lektion 10 und Lektion 24). In

Zeile 330 erfolgt eine Abfrage, ob die zwei rechts befindlichen Zeichen der System-Variablen TI$ ungleich dem Inhalt von AZ$ sind - dies trifft zu, wenn die Uhrzeit eine Sekunde vorangeschritten ist. Wenn Ungleichheit festgestellt wird, erfolgt ein Sprung zur Zeile 150; andernfalls wird automatisch der nächste Befehl (vgl. hierzu Lektion 18) in

Zeile 340 ausgeführt, mit dem ein Sprung zur Abfrage in Zeile 330 erfolgt. Anders ausgedrückt: Die Befehle in den Zeilen 330 und 340 werden so häufig wiederholt, bis die Uhrzeit 1 Sekunde vorangeschritten ist.

Nachdem wir das erläuterte Programm mit dem Kommando

SAVE "UHRZEIT1"

auf Bandkassette gespeichert haben, wollen wir es dahingehend erweitern, daß zu jeder vollen Stunde die Melodie "Horch, es tönt der Glockenton . . . " erklingt.

Programm "UHRZEIT2"

Zunächst ändern wir die Zeile 20:

```
20 REM * UHRZEIT2 *
```

Anschließend ergänzen wir zwischen der Zeile 250 und Zeile 260 die folgenden Anweisungen, mit denen untersucht wird, ob eine "volle Stunde" (also: MMSS = "0000") eingetreten ist:

```
251 REM
252 REM VOLLE STUNDE?
253 REM -------------
254 LET VS$ = RIGHT$(TI$ , 4)
255 IF VS$ = "0000" THEN GOSUB 1000
```

Aufgrund der Anweisung in Zeile 255 wird in das Unterprogramm UP-MUSIK (ab Zeile 1000) verzweigt (vgl. hierzu Lektion 21), wenn VS$ = "0000" ist:

```
1000 REM ************
1010 REM * UP-MUSIK *
1020 REM ************
1030 TEMPO 7
1040 MUSIC "C8DEFEDCEFGAGFE"
1050 MUSIC "+C9R+C9R+C9R+C9"
1060 RETURN
```

Durch eine Änderung der Befehle in den Zeilen 254 und 255 kann man die Melodie zu einem beliebigen anderen Zeitpunkt ertönen lassen. (Wir erinnern uns: Die Änderung bzw. Ergänzung des Programms kann unabhängig von der jeweiligen "Situation" auf dem Bildschirm erfolgen. Aufgrund der angegebenen Zeilennummer wird die Programmzeile im Arbeitsspeicher an der vorgesehenen Stelle eingefügt bzw. eine überholte Programmzeile überschrieben. - Mit Hilfe des Systemkommandos

L I S T können wir uns jederzeit eine Übersicht über die aktuelle Situation im Arbeitsspeicher verschaffen.)

Wen es nun noch stören sollte, daß das Programm an einigen Stellen nicht in Zehnerschritten numeriert ist, kann diesem Mangel mit Hilfe des Systemkommandos

RENUM

abhelfen (vgl. hierzu Seite Anhang-14 ff.). Dadurch werden nicht nur die Zeilennummern, sondern auch die entsprechenden Sprungadressen in der für BASIC üblichen Weise renumeriert:

```
10 REM ***********
20 REM * UHRZEIT2 *
30 REM ***********
40 CLS
50 REM EINGABE DER AKTUELLEN UHRZEIT
60 REM ---------------------------
70 PRINT : PRINT
80 PRINT "BITTE UHRZEIT EINGEBEN!"
90 PRINT
100 PRINT "FORM: HHMMSS"
110 PRINT
120 PRINT TAB(4);
130 INPUT UZ$
140 TI$ = UZ$
150 CLS
160 PRINT : PRINT
170 REM AUSGABE DER UHRZEIT
180 REM ------------------
190 PRINT TAB(5);
200 PRINT LEFT$(TI$, 2);
210 PRINT " UHR ";
220 PRINT MID$(TI$, 3, 2);
230 PRINT " MINUTEN ";
240 PRINT RIGHT$(TI$, 2);
250 PRINT " SEKUNDEN"
260 REM
270 REM VOLLE STUNDE?
280 REM -------------
290 LET VS$ = RIGHT$(TI$ , 4)
300 IF VS$ = "0000" THEN GOSUB 400
```

```
310 REM
320 REM WARTESCHLEIFE, BIS SEKUNDEN-
330 REM ZEIGEER 1 SEKUNDE VORGERUECKT
340 REM IST - DANN SPRUNG
350 REM NACH ZEILE 150
360 REM ----------------------------
370 LET AZ$ = RIGHT$(TI$, 2)
380 IF RIGHT$(TI$, 2) <> AZ$ THEN 150
390 GOTO 380
400 REM ************
410 REM * UP-MUSIK *
420 REM ************
430 TEMPO 7
440 MUSIC "C8DEFEDCEFGAGFE"
450 MUSIC "+C9R+C9R+C9R+C9"
460 RETURN
```

9. Bemerkungen mit REM

LE 9

Nachdem wir uns ausführlich mit den Konstanten und Variablen beschäftigt haben, also dem "W O M I T - Teil" der Befehle, wollen wir uns nun dem Operationsteil ("W A S - Teil") der Anweisungen zuwenden.

Die Befehle, die wir fast immer zu Beginn unserer Programme aufführen (aber auch "mittendrin" zu finden sind), wie beispielsweise

```
10 REM ***********
20 REM * ALARICH *
30 REM ***********
```

sind im strengen Sinne keine Instruktionen, denn während der Programmausführung verursachen sie keine eigentlichen Verarbeitungsschritte.

Die Anweisung

REM (Abkürzung für engl. "remark" = Bemerkung)

dient nur der Kennzeichnung und Erläuterung von Programmen oder Programmteilen und erhöht damit deren Lesbarkeit und Überschaubarkeit. Der REM-Befehl kann an beliebigen Stellen des Programms aufgeführt werden.

PE 9

Welche der folgenden Behauptungen ist/sind richtig?

A Die REM-Anweisung dient zur Durchführung von arithmetischen Operationen.

B Mit der REM-Anweisung ist es möglich, Bemerkungen in ein Programm einzufügen.

C Die REM-Anweisung darf nur zu Beginn eines Programms aufgeführt werden.

D Die REM-Anweisung kann an beliebigen Stellen des Programms aufgeführt werden.

E Mit der REM-Anweisung setzt man numerische Variablen auf 0 (Null).

Der/die Lösungsbuchstabe/n lautet/lauten

B, D.......................... .

LÖS 9

Die Lösungsbuchstaben lauten

B, D .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 9**!

ÜBUNGsprogramm zu Lektion 9

Programmbeschreibung (Weltzeitenuhr):

Wir wollen das Programm "UHRZEIT2" aus Lektion 8 wieder aufgreifen und zu einer "Weltzeitenuhr" umgestalten. - Die jeweilige lokale Uhrzeit der folgenden Orte mit den angegebenen Zeitverschiebungen soll aufgeführt werden:

(Zeitveränderungen aufgrund gesetzlicher Bestimmungen - die sog. Sommerzeiten - sind hier nicht berücksichtigt. Sie lassen sich jedoch bei Bedarf durch einfache Änderung der Zu- bzw. Abschläge einführen.)

Ort	Zu-/Abschlag	Uhrzeit (Beispiel)
Los Angeles	- 8	2
Chikago	- 7	3
New York	- 6	4
Rio de Janeiro	- 4	6
Dakar	- 2	8
** Hamburg **	0	10
Kapstadt	+ 1	11
Moskau	+ 2	12
Hongkong	+ 7	17
Tokio	+ 8	18
Sydney	+ 9	19

Entwurf des Bildschirmbildes:

Um eine optimale Ausgabe zu erzielen, ist es empfehlenswert, mit Hilfe eines einfach zu gestaltenden Bildschirmentwurfformulars das gewünschte Bildschirmbild zu planen (vgl. hierzu die folgende Seite).

Änderungen und Ergänzungen:

Ausgehend von unserem Programm "UHRZEIT2" ändern wir zunächst Zeile 20:

```
20 REM * UHRZEIT3 *
```

Anschließend löschen wir die Zeilen 160 - 250 mit dem Systemkommando

```
DELETE 160 - 250
```

und verschieben die Befehle, die ab Zeile 260 beginnen, dahingehend, daß sie mit Zeile 3000 anfangen (um ausreichenden Zwischenraum für einzufügende Anweisungen zu gewinnen), und zwar mit dem Systemkommando

```
RENUM 3000, 260
```

Nun ergänzen wir die Instruktionen der Zeilen 160 - 1430, ändern den Sprungbefehl der (renumerierten) Zeile 3120 (alte Zeile 380) dergestalt ab, daß als Sprungziel Zeile 410 (statt 150) angegeben wird. (Außerdem ist die REM-Anweisung in Zeile 3090 zu aktualisieren!) Abschließend wird mit dem Systemkommando

```
RENUM
```

eine lückenlose Zeilennumerierung in Zehnerschritten vorgenommen. Wir erhalten dann das folgende Programm:

(vgl. Seite 9-08 ff.)

```
               STUNDEN MINUTEN SEKUNDEN

LOS ANGELES      XX      XX      XX

CHICAGO          XX      XX      XX

NEW YORK         XX      XX      XX

RIO DE JANEIRO   XX      XX      XX

DAKAR            XX      XX      XX

** HAMBURG **    XX      XX      XX

KAPSTADT         XX      XX      XX

MOSKAU           XX      XX      XX

HONGKONG         XX      XX      XX

TOKIO            XX      XX      XX

SYDNEY           XX      XX      XX


```

Programmliste:

```
10 REM ***********
20 REM * UHRZEIT3 *
30 REM ***********
40 CLS
50 REM EINGABE DER AKTUELLEN UHRZEIT
60 REM ----------------------------
70 PRINT : PRINT
80 PRINT "BITTE UHRZEIT EINGEBEN!"
90 PRINT
100 PRINT "FORM: HHMMSS"
110 PRINT
120 PRINT TAB(4);
130 INPUT UZ$
140 TI$ = UZ$
150 CLS
160 REM AUSGABE TEXT
170 REM -----------
180 PRINT TAB(15) "STUNDEN MINUTEN SEKUN
DEN"
190 PRINT
200 PRINT "LOS ANGELES"
210 PRINT
220 PRINT "CHICAGO"
230 PRINT
240 PRINT "NEW YORK"
250 PRINT
260 PRINT "RIO DE JANEIRO"
270 PRINT
280 PRINT "DAKAR"
290 PRINT
300 PRINT "** HAMBURG **"
310 PRINT
320 PRINT "KAPSTADT"
330 PRINT
340 PRINT "MOSKAU"
350 PRINT
360 PRINT "HONGKONG"
370 PRINT
380 PRINT "TOKIO"
390 PRINT
400 PRINT "SYDNEY"
410 REM
```

```
420 REM ------------------------
430 LET ST = VAL(LEFT$(TI$, 2))
440 LET MI$ = MID$(TI$, 3, 2)
450 LET SE$ = RIGHT$(TI$, 2)
460 REM ------------------------
470 REM * UHRZEIT LOS ANGELES *
480 LET LA = ST - 8
490 IF LA < 0 THEN LET LA = LA + 24
500 LET LA$ = STR$(LA)
510 IF LEN(LA$)=1 THEN LET LA$=" "+LA$
520 CURSOR 17, 2
530 PRINT LA$
540 CURSOR 25, 2
550 PRINT MI$
560 CURSOR 33, 2
570 PRINT SE$
580 REM * UHRZEIT CHICAGO *
590 LET CH = ST - 7
600 IF CH < 0 THEN LET CH = CH + 24
610 LET CH$ = STR$(CH)
620 IF LEN(CH$)=1 THEN LET CH$=" "+CH$
630 CURSOR 17, 4
640 PRINT CH$
650 CURSOR 25, 4
660 PRINT MI$
670 CURSOR 33, 4
680 PRINT SE$
690 REM * UHRZEIT NEW YORK *
700 LET NY = ST - 6
710 IF NY < 0 THEN LET NY = NY + 24
720 LET NY$ = STR$(NY)
730 IF LEN(NY$)=1 THEN LET NY$=" "+NY$
740 CURSOR 17, 6
750 PRINT NY$
760 CURSOR 25, 6
770 PRINT MI$
780 CURSOR 33, 6
790 PRINT SE$
```

```
800 REM * UHRZEIT RIO DE JANEIRO *
810 LET RD = ST - 4
820 IF RD < 0 THEN LET RD = RD + 24
830 LET RD$ = STR$(RD)
840 IF LEN(RD$)=1 THEN LET RD$=" "+RD$
850 CURSOR 17, 8
860 PRINT RD$
870 CURSOR 25, 8
880 PRINT MI$
890 CURSOR 33, 8
900 PRINT SE$
910 REM * UHRZEIT DAKAR *
920 LET DA = ST - 2
930 IF DA < 0 THEN LET DA = DA + 24
940 LET DA$ = STR$(DA)
950 IF LEN(DA$)=1 THEN LET DA$=" "+DA$
960 CURSOR 17, 10
970 PRINT DA$
980 CURSOR 25, 10
990 PRINT MI$
1000 CURSOR 33, 10
1010 PRINT SE$
1020 REM * UHRZEIT  H A M BU R G  *
1030 LET ST$ = STR$(ST)
1040 IF LEN(ST$)=1 THEN LET ST$=" "+ST$
1050 CURSOR 17, 12
1060 PRINT ST$
1070 CURSOR 25, 12
1080 PRINT MI$
1090 CURSOR 33, 12
1100 PRINT SE$
1110 REM * UHRZEIT KAPSTADT *
1120 LET KA = ST + 1
1130 IF KA > 24 THEN LET KA = KA - 24
1140 LET KA$ = STR$(KA)
1150 IF LEN(KA$)=1 THEN LET KA$=" "+KA$
1160 CURSOR 17, 14
1170 PRINT KA$
1180 CURSOR 25, 14
1190 PRINT MI$
1200 CURSOR 33, 14
1210 PRINT SE$
```

```
1220 REM * UHRZEIT MOSKAU *
1230 LET MO = ST + 2
1240 IF MO > 24 THEN LET MO = MO - 24
1250 LET MO$ = STR$(MO)
1260 IF LEN(MO$)=1 THEN LET MO$=" "+MO$
1270 CURSOR 17, 16
1280 PRINT MO$
1290 CURSOR 25, 16
1300 PRINT MI$
1310 CURSOR 33, 16
1320 PRINT SE$
1330 REM * UHRZEIT HONGKONG *
1340 LET HO = ST + 7
1350 IF HO > 24 THEN LET HO = HO - 24
1360 LET HO$ = STR$(HO)
1370 IF LEN(HO$)=1 THEN LET HO$=" "+HO$
1380 CURSOR 17, 18
1390 PRINT HO$
1400 CURSOR 25, 18
1410 PRINT MI$
1420 CURSOR 33, 18
1430 PRINT SE$
1440 REM * UHRZEIT TOKIO *
1450 LET TK = ST + 8
1460 IF TK > 24 THEN LET TK = TK - 24
1470 LET TK$ = STR$(TK)
1480 IF LEN(TK$)=1 THEN LET TK$=" "+TK$
1490 CURSOR 17, 20
1500 PRINT TK$
1510 CURSOR 25, 20
1520 PRINT MI$
1530 CURSOR 33, 20
1540 PRINT SE$
1550 REM * UHRZEIT SYDNEY *
1560 LET SY = ST + 9
1570 IF SY > 24 THEN LET SY = SY - 24
1580 LET SY$ = STR$(SY)
1590 IF LEN(SY$)=1 THEN LET SY$=" "+SY$
1600 CURSOR 17, 22
1610 PRINT SY$
1620 CURSOR 25, 22
1630 PRINT MI$
1640 CURSOR 33, 22
1650 PRINT SE$
1660 REM
```

```
1670 REM VOLLE STUNDE?
1680 REM -------------
1690 LET VS$ = RIGHT$(TI$ , 4)
1700 IF VS$ = "0000" THEN GOSUB 1800
1710 REM
1720 REM WARTESCHLEIFE, BIS SEKUNDEN-
1730 REM ZEIGEER 1 SEKUNDE VORGERUECKT
1740 REM IST - DANN SPRUNG
1750 REM NACH ZEILE 410
1760 REM -----------------------------
1770 LET AZ$ = RIGHT$(TI$, 2)
1780 IF RIGHT$(TI$, 2) <> AZ$ THEN 410
1790 GOTO 1780
1800 REM ************
1810 REM * UP-MUSIK *
1820 REM ************
1830 TEMPO 7
1840 MUSIC "C8DEFEDCEFGAGFE"
1850 MUSIC "+C9R+C9R+C9R+C9"
1860 RETURN
```

10. Zuweisungen mit LET

LE 10

Mit dem Befehl

LET (von engl. "to let" = lassen)

wird der Variablen links vom Gleichheitszeichen der Wert des Ausdrucks zugewiesen, der rechts vom Gleichheitszeichen steht.

Das Gleichheitszeichen (=) bedeutet in der LET-Anweisung nicht "ist gleich", sondern "ergibt sich aus" oder "(ist) soviel wie".

Beispiele:

(1) 40 LET A = 10
bedeutet:
Laß A sein soviel wie 10

(2) 50 LET A = A + 1
bedeutet:
A (neuer Wert) ergibt sich aus A (alter Wert) + 1

(3) 60 LET B = A * A
bedeutet:
B ergibt sich aus A x A

(4) 300 LET C = SQR(B + A)

bedeutet:

Laß C sein soviel wie die Quadratwurzel von (B + A)

(Anmerkung: SQR ist eine Abkürzug für engl. "square root" = Quadratwurzel. Es handelt sich dabei um eine von zahlreichen sog. vordefinierten Funktionen, die BASIC zur Verfügung stellt. Man veranlaßt damit die Berechnung der Quadratwurzel des Klammerausdrucks, der auf SQR folgt. Wenn beispielsweise A den Wert 9 enthalten würde und B den Wert 40, dann wäre (A + B) = 49 und SQR(A + B) = 7, also erhielte C im obigen Befehl den Wert 7 zugewiesen.)

Bei den meisten BASIC-Rechnern - wie auch beim SHARP MZ-700 - ist es heute zulässig, im Rahmen des Zuweisungsbefehls das Element "LET" fortzulassen.

Beispiel:

statt: 300 LET C = SQR(X - Y)

auch zulässig: 300 C = SQR(X - Y)

Damit die Zuweisung optisch besonders deutlich ist, sollte man aber im Regelfall auf "LET" nicht verzichten.

MERKE:

Bei der LET-Anweisung muß links vom Gleichheitszeichen immer ein Variablenname stehen und niemals ein zusammengesetzter Ausdruck.

PE 10

(Bitte, die linke Seite abdecken!)

(1) Welche der folgenden Behauptungen ist/sind richtig?

A Mit der LET-Anweisung wird der Variablen links vom Gleichheitszeichen der Wert des Ausdrucks zugewiesen, der rechts vom Gleichheitszeichen steht.

B Bei der LET-Anweisung steht links vom Gleichheitszeichen ein Variablenname oder ein zusammengesetzter Ausdruck.

C Bei der LET-Anweisung muß links vom Gleichheitszeichen immer ein Variablenname stehen und niemals ein zusammengesetzter Ausdruck.

Der/die Lösungsbuchstabe/n lautet/lauten A, C . . . .

(2) Wie lautet die vordefinierte Funktion, mit der man in BASIC eine Quadratwurzel berechnet?

Die vordefinierte Funktion lautet SQR . . . .

(3) Beispielprogramm:

```
10 LET A = 10
20 LET B = 13
30 LET C = A + B
```

Wie lautet nach Ausführung des Programms der Inhalt der Variablen C ?

Inhalt der Variablen C (nach Programmausführung):

23 . . . .

LÖS 10

(1) Die Lösungsbuchstaben lauten A, C .

(2) Die vordefinierte Funktion, mit der man in BASIC Quadratwurzeln berechnet, lautet

SQR(X) .

(3) Beispielprogramm:

```
10 LET A = 10
20 LET B = 13
30 LET C = A + B
```

Inhalt der Variablen C (nach Programmausführung):

23 .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 10** !

ÜBUNGsprogramm zu Lektion 10

Problemstellung / Programmbeschreibung:

Man erzählt sich die Anekdote, daß der Schah vom Schach so begeistert war, daß er dem Erfinder des Spiels einen Wunsch gewährte. Der Untertan erbat sich die Anzahl der Reiskörner, die auf dem 64. Feld des Schachbretts liegen würden, wenn man beim ersten Feld mit einem Korn begänne und eine Verdopplung von Feld zu Feld vornähme; also

Feld 1 entspräche 1 Korn,
Feld 2 entspräche 2 Körnern,
Feld 3 entspräche 4 Körnern,
Feld 4 entspräche 8 Körnern,
.
.
.
Feld 64 entspräche ?? Körnern.

Es wird berichtet, daß der Schah die Erfüllung des Wunsches umgehend zusagte - so leichtsinnig wäre er sicherlich nicht gewesen, wenn er zur Berechnung der Menge einen SHARP MZ-700 gehabt hätte.

Mit dem folgenden Programm kann man die Anzahl der Körner berechnen, die auf dem 64. Feld zu liegen kämen.

Programmliste (Alternative 1):

```
10 REM ***********************
20 REM * REISKOERNER FUER DIE *
30 REM *    ERFINDUNG DES     *
40 REM *    SCHACHSPIELS      *
50 REM *                      *
60 REM *    ALTERNATIVE 1     *
70 REM ***********************
80 CLS
90 LET A = 1
100 LET Z = 1
110 PRINT "FELD"; Z; ":"; A; " KOERNER"
120 LET A = A * 2
130 LET Z = Z + 1
140 IF Z > 64 THEN 160
150 GOTO 110
160 END
```

Hinweise und Erläuterungen:

(1) Die Ausgabe auf dem Bildschirm läßt sich durch Niederdrücken der BREAK-Taste unterbrechen.

(2) Die Befehle der Zeilen 110 - 150 bilden eine Schleife, deren Endebedingung mit einer IF . . -Anweisung überprüft wird. (Vgl. hierzu die Ausführungen in Lektion 18!)

(3) Das Ergebnis ergibt eine Gleitkommazahl. Zu deren Interpretation verweisen wir auf die Ausführungen auf Seite 8-02!

Programmliste (Alternative 2):

```
10 REM ***********************
20 REM * REISKOERNER FUER DIE *
30 REM *    ERFINDUNG DES     *
40 REM *    SCHACHSPIELS      *
50 REM *                      *
60 REM *    ALTERNATIVE 2     *
70 REM ***********************
80 CLS
90 LET A = 1
```

```
100 FOR Z = 1 TO 64
110 PRINT "FELD"; Z; ":"; A; " KOERNER"
120 LET A = A * 2
130 NEXT Z
140 END
```

Hinweise und Erläuterungen:

Im Programm der Alternative 2 wurde die Schleife mit FOR .. NEXT codiert. (Vgl. hierzu die Ausführungen in Lektion 20!)

Eine etwas andere Form der Anekdote besagt, daß der Untertan (nicht "nur" die Körner des 64. Feldes, sondern) die Summe aller Körner von Feld 1 bis Feld 64 erhielt. Die Berechnung dieser Anzahl ergibt sich aus der Alternative 3.

Programmliste (Alternative 3):

```
10 REM ***********************
20 REM * REISKOERNER FUER DIE *
30 REM *    ERFINDUNG DES     *
40 REM *    SCHACHSPIELS      *
50 REM *                      *
60 REM *    ALTERNATIVE 3     *
70 REM ***********************
80 CLS
90 LET A = 1
100 FOR Z = 1 TO 64
110 LET S = S + A
120 PRINT "FUER FELD"; Z
130 PRINT "ANZAHL DER KOERNER:"; A
140 PRINT "SUMME  DER KOERNER:"; S
150 PRINT
160 LET A = A * 2
170 NEXT Z
180 END
```

11. Arithmetische Operatoren

LE 11.1

In unseren bisherigen Beispielen haben wir schon einige arithmetische Operatoren verwendet, die wir nun in systematischer Vollständigkeit vorstellen wollen:

Operator	Bedeutung	Beispiel	mathematische Schreibweise
↑	Potenzierung	M ↑ A	M^A
*	Multiplikation	J * A	J · A oder auch J x A
/	Division	H / A	H : A oder auch $\frac{H}{A}$
+	Addition	M + A	M + A
-	Subtraktion	N - M	N - M

Die obigen arithmetischen Operatoren werden in Verbindung mit numerischen Variablen, numerischen Konstanten, Funktionen wie SQR(X) und Klammern benutzt, um mathematische Ausdrücke zu bilden.

Bei der Auswertung eines Ausdrucks wendet BASIC jene Reihenfolgeregeln an, die auch in der Mathematik gültig sind:

(höchste Priorität)	Priorität 1:	Klammerausdrücke
	Priorität 2:	Funktionen[1]
	Priorität 3:	Potenzierung
	Priorität 4:	Multiplikation / Division
(niedrigste Priorität)	Priorität 5:	Addition / Subtraktion

Bei Gleichrangigkeit rechnet BASIC von links nach rechts.

Beispiel:

(Um besonders anschaulich zu sein, sind in dem zu berechnenden Ausdruck nur Konstanten aufgeführt. Das errechnete Ergebnis soll der Variablen E1 zugewiesen werden.)

```
100 LET E1 = (7 - 3) + SQR(30 - 5) - 5↑3 + 6 - 2 * 4 / 2
```

Die Reihenfolge für die Bearbeitung des Ausdrucks ergibt sich aus der Darstellung auf der folgenden Seite. (Für die Wiedergabe des obigen Befehls benötigt man mehr als eine Bildschirmzeile. Wir wollen deshalb nochmals in Erinnerung rufen, daß erst nach dem Drücken der [CR]-Taste die gesamte Anweisung in den Arbeitsspeicher übertragen wird.)

1) Einzelheiten hierzu in den Lektionen 24 und 25

Reihenfolge der Bearbeitung (durch Unterstreichung gekennzeichnet)

```
(7 - 3) + SQR(30 - 5) - 5↑3 + 6 - 2 * 4 / 2          1.
4       + SQR(30 - 5) - 5↑3 + 6 - 2 * 4 / 2          2.
4       + SQR(25)     - 5↑3 + 6 - 2 * 4 / 2          3.
4       +   5         - 5↑3 + 6 - 2 * 4 / 2          4.
4       +   5         - 125 + 6 - 2 * 4 / 2          5.
4       +   5         - 125 + 6 -  8    / 2          6.
4       +   5         - 125 + 6 -     4              7.
        9             - 125 + 6 -     4              8.
            - 116           + 6 -     4              9.
                      - 110     -     4              10.
                             - 114
```

Wenn man die Rangordnung der Operatoren vergessen haben sollte, so kann man die gewünschte Reihenfolge für die Bearbeitung eines Ausdrucks durch Verwendung von Klammern sicherstellen. Überflüssige Klammern, d. h. solche, die eine durch die Rangfolge der Operatoren vorgegebene Verarbeitungsreihenfolge nicht ändern, sind unschädlich.

(A * B) + (C * D) ist also gleichwertig dem Ausdruck

A * B + C * D .

Bei sehr vielen BASIC-Rechnern sind zwei direkt aufeinanderfolgende Operatoren verboten. Sie müssen dann durch Klammern voneinander getrennt werden.

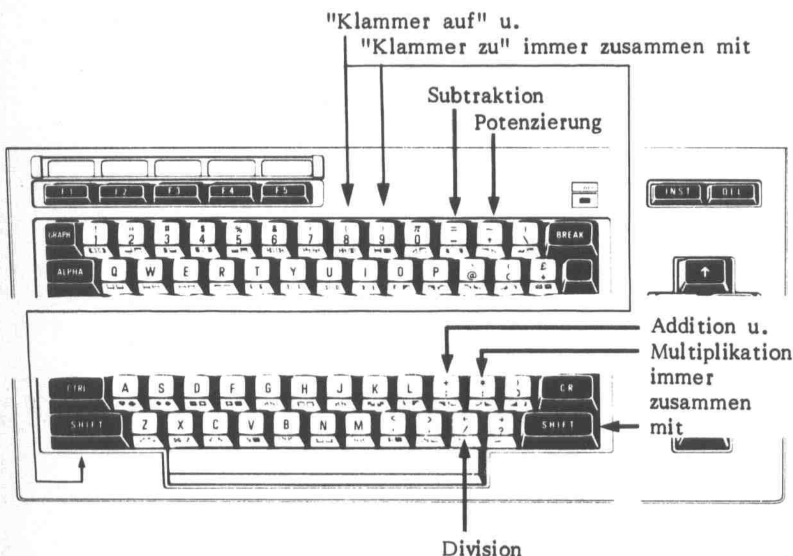
Beispiele:

(1) A / -8 ▶ bei vielen BASIC-Systemen unzulässig

(2) A / (-8) ▶ immer zulässig

Die oben dargestellte Einschränkung gilt nicht beim SHARP MZ-700; für ihn ist sowohl die Schreibweise (1) als auch die Darstellung (2) korrekt!

Die arithmetischen Operatoren (u. Klammern) auf der Tastatur des SHARP MZ-700:



Ergänzender Hinweis für den fortgeschrittenen Leser:

Wir wollen hier noch kurz auf die Bearbeitungspriorität des "monadischen Minus-Operators" eingehen.

Was ist ein "monadischer Minus-Operator"?

Ein monadischer Minus-Operator ist ein Minuszeichen, dem keine Konstante oder Variable vorangeht (Beispiel: 10 LET A = -5). Hinsichtlich der Bearbeitungspriorität wird der monadische Minus-Operator bei BASIC-Systemen verschiedener Hersteller n i c h t einheitlich behandelt. Der SHARP MZ-700 vollzieht die monadische Minus-Operation unmittelbar nach dem Potenzieren.

Beispiel:

```
10 LET A = -5
20 LET B = -A↑2
30 PRINT B
RUN
-25
```

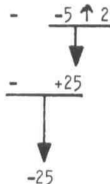
Reihenfolge der Berechnung:

Der zu berechnende Ausdruck lautet: - A ↑ 2

Da die Variable A den Inhalt -5 besitzt (vgl. Zeile 10 !), könnte man den zu berechnenden Ausdruck auch folgendermaßen darstellen:

1. Bearbeitungsschritt: Der Inhalt der Variablen A wird potenziert:

2. Bearbeitungsschritt: Der monadische Minus-Operator kommt zur Ausführung:



PE 11.1

(1) Vervollständigen Sie die folgende Übersicht!

Operator	Bedeutung	Beispiel (BASIC-Schreibweise)
.....	Potenzierung	....
.....	Multiplikation	....
.....	Division	....
.....	Addition	....
.....	Subtraktion	....

(2) Geben Sie die Reihenfolge an, in der mathematische Ausdrücke berechnet werden (1 = höchste Priorität, 5 = niedrigste Priorität)!

	Priorität
Addition	.....
Subtraktion	.....
Klammerausdrücke	.....
Funktionen	.....
Potenzierung	.....
Multiplikation	.....
Division	.....

LÖS 11.1

(1) Die vervollständigte Übersicht hat folgendes Aussehen:

Operator	Bedeutung	Beispiel (BASIC-Schreibweise)
↑	Potenzierung	E ↑ L
*	Multiplikation	S * B
/	Division	E / T
+	Addition	H + 13
-	Subtraktion	10 - J

(2) Bei der Auswertung eines mathematischen Ausdrucks wendet BASIC die folgende Reihenfolge an:

	Priorität		
Addition	5	(wie Subtraktion)	← 1)
Subtraktion	5	(wie Addition)	← 1)
Klammerausdrücke	1		← 2)
Funktionen	2		
Potenzierung	3		
Multiplikation	4	(wie Division)	
Division	4	(wie Multiplikation)	

1) niedrigste Priorität
2) höchste Priorität

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 11.1** !

LE 11.2

Es sind vor allem zwei Umstände, die die Unterschiede zwischen der mathematischen Schreibweise einerseits und der BASIC-Schreibweise andererseits begründen:

- Die Operatoren haben z. T. ein anderes Aussehen.
- Um die in der mathematischen Schreibweise ausgedrückte Rangfolge zu erhalten, können in BASIC zusätzliche Klammern erforderlich sein.

Beispiele:

	Mathematische Schreibweise		BASIC-Schreibweise
(1)	$\frac{A + H}{N + C}$		(A + H) / (N + C)
(2)	$\frac{H + A}{R}$		(H + A) / R
(3)	$\frac{I + M}{-H}$		(I + M) / (-H)
		beim SHARP MZ-700 auch zulässig:	(I + M) / -H
(4)	$\frac{A \cdot M}{N}$		A * M / N
(5)	$\frac{\frac{X}{Y}}{Z}$		X / Y / Z

	Mathematische Schreibweise	BASIC-Schreibweise
(6)	$\frac{X}{\frac{A}{B}}$	X / (A / B)
(7)	$(A \cdot B)^P \cdot C$	(A * B)↑P * C
(8)	X^{Y^Z}	X↑(Y↑Z)

PE 11.2

(Bitte, die linke Seite abdecken!)

Vervollständigen Sie die folgende Übersicht!

	Mathematische Schreibweise	BASIC-Schreibweise
(1)	$X + 2Y$	X+2*Y ....
(2)	$A + \frac{B}{C}$	A+B/C ....
(3)	$\frac{A + B + C}{D}$	(A+B+C)/D ....
(4)	$\frac{C + H}{I - K}$	(C+H)/(I-K) ....
(5)	$\frac{E + L}{-S}$	(E+L)/(-S) ....
(6)	$\frac{\frac{A}{B}}{C}$	(A/B)/C ....
(7)	$\frac{A}{\frac{B}{C}}$	A/(B/C) ....

LÖS 11.2

Die vervollständigte Übersicht hat folgendes Aussehen:

	Mathematische Schreibweise	BASIC-Schreibweise
(1)	$X + 2Y$	X + 2 * Y
(2)	$A + \frac{B}{C}$	A + B / C
(3)	$\frac{A + B + C}{D}$	(A + B + C) / D
(4)	$\frac{C + H}{I - K}$	(C + H) / (I - K)
(5)	$\frac{E + L}{-S}$	(E + L) / (-S)
(6)	$\frac{\frac{A}{B}}{C}$	A / B / C
(7)	$\frac{A}{\frac{B}{C}}$	A / (B / C)

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 11.1** !

ÜBUNGsprogramm zu Lektion 11

Programmbeschreibung (Reaktionstest):

Mit diesem Programm kann man das menschliche Reaktionsvermögen messen.

Nach der Eingabe der Anzahl der Teilnehmer wird jede Person fünfmal dem folgenden Test unterworfen:

Auf dem Bildschirm erscheinen - in zeitlich unterschiedlicher Geschwindigkeit - die folgenden Worte:

- AUF DIE PLAETZE!
- FERTIG!
- LOS!

Nach der Ausgabe des letzten Wortes ("LOS!") hat der Bediener so schnell wie möglich eine beliebige Taste zu drücken, worauf der Rechner die Reaktionszeit angibt. Wer eine Taste permanent niedergedrückt hält, um so ein optimales Ergebnis zu erzielen, erhält eine volle "Straf"-Sekunde zugewiesen.

Nachdem jeder Spieler fünfmal den Test durchlaufen hat, wird abschließend für jeden die durchschnittliche Reaktionszeit ausgegeben und mit einer Zensur versehen.

Programmliste:

```
10 REM *************
20 REM * REAKTIONS- *
30 REM *    TEST    *
40 REM *************
50 REM
60 REM BILD-1
70 REM ------
80 CLS
90 PRINT : PRINT : PRINT
100 PRINT TAB(8) "****************"
110 PRINT TAB(8) "*              *"
120 PRINT TAB(8) "* REAKTIONSTEST *"
130 PRINT TAB(8) "*              *"
140 PRINT TAB(8) "****************"
150 FOR I = 1 TO 6 : PRINT : NEXT I
160 PRINT TAB(8) "TEILNEHMERZAHL";
170 INPUT Z
180 FOR I = 1 TO 6 : PRINT :: NEXT I
190 PRINT TAB(20) "'W'-TASTE DRUECKEN!"
200 GET A$
210 IF A$ <> "W" THEN 200
220 REM
230 REM DIMENSIONIERUNG EINES RECHEN-
240 REM FELDES ENTSPRECHEND DER
250 REM TEILNEHMERZAHL
260 REM ---------------------------
270 DIM RF(Z)
280 FOR X = 1 TO Z
290 REM
300 REM BILD-2
310 REM ------
320 CLS
330 PRINT
340 PRINT "SEHR GEEHRTER TEILNEHMER NUMM
ER"; X; " !"
350 PRINT : PRINT
360 PRINT "NACH DEM DRUECKEN DER 'W'-TAS
TE"
370 PRINT
380 PRINT "ERSCHEINT 5 X DAS KOMMANDO"
390 PRINT
```

```
400 PRINT "- AUF DIE PLAETZE!"
410 PRINT
420 PRINT "- FERTIG!"
430 PRINT
440 PRINT "- LOS!"
450 PRINT : PRINT
460 PRINT "DRUECKEN SIE DANN JEWEILS SO
SCHNELL"
470 PRINT
480 PRINT "WIE MOEGLICH EINE BELIEBIGE"
490 PRINT
500 PRINT "ZEICHENTASTE!"
510 PRINT : PRINT : PRINT
520 PRINT TAB(20) "'W'-TASTE DRUECKEN!"
530 GET A$
540 IF A$ <> "W" THEN 530
550 REM
560 REM BILD-3
570 REM ------
580 FOR Z5 = 1 TO 5
590 CLS
600 LET S = 0
610 PRINT : PRINT : PRINT
620 PRINT TAB(10) "AUF DIE PLAETZE!"
630 PRINT : PRINT : PRINT
640 FOR I = 1 TO RND(1)*3000 : NEXT I
650 PRINT TAB(10) "FERTIG!"
660 PRINT : PRINT : PRINT
670 FOR I = 1 TO RND(1)*3000 : NEXT I
680 PRINT TAB(10) "LOS!"
690 GET A$
700 IF A$ <> "" THEN 730
710 LET S = S + 1
720 GOTO 690
730 REM BILD-4
740 REM ------
750 CLS
760 IF S > 0 THEN 880
770 REM VERGABE VON STRAFPUNKTEN
780 REM -----------------------
790 PRINT
800 PRINT "SIE HABEN VORZEITIG EINE"
810 PRINT
820 PRINT "TASTE BETAETIGT!"
830 PRINT : PRINT : PRINT
```

```
840 PRINT "SIE ERHALTEN DAHER ALS REAKTI
ONSZEIT"
850 PRINT
860 PRINT "1 'STRAF'-SEKUNDE!"
870 LET S = 7060 / 60
880 REM AUSWERTUNG UND AUSGABE DER
890 REM EINZELNEN REAKTIONS-
900 REM GESCHWINDIGKEITEN
910 REM ------------------------
920 LET RF(X) = RF(X) + S
930 PRINT
940 PRINT "IHRE REAKTIONSZEIT BETRUG"
950 PRINT
960 PRINT S * 60 / 7060
970 PRINT
980 PRINT "SEKUNDEN."
990 PRINT : PRINT : PRINT
1000 PRINT TAB(20) "'W'-TASTE!"
1010 GET A$
1020 IF A$ <> "W" THEN 1010
1030 NEXT Z5
1040 NEXT X
1050 REM BILD-5 (GESAMTERGEBNIS)
1060 REM ----------------------
1070 CLS
1080 PRINT
1090 PRINT "DURCHSCHNITTSWERTE ALLER TEI
LNEHMER:"
1100 PRINT "----------------------------
--------"
1110 PRINT
1120 FOR I = 1 TO Z
1130 LET E = RF(I) * 60 / 7060 / 5
1140 IF E < .2 THEN LET E$ = "(= SEHR GU
T) " : GOTO 1200
1150 IF E < .225 THEN LET E$ = "(= GUT)"
 : GOTO 1200
1160 IF E < .25 THEN LET E$ = "(= BEFRIE
DIGEND)" : GOTO 1200
1170 IF E < .275 THEN LET E$ = "(= AUSRE
ICHEND)" : GOTO 1200
1180 IF E < .3 THEN LET E$ = "(= MANGELH
AFT)" : GOTO 1200
```

```
1190 LET E$ = "(= VERKALKT?)"
1200 PRINT "TEILN."; I; ":"; E; " SEK. "
; E$
1210 PRINT
1220 NEXT I
1230 END
```

Erläuterungen:

(1) Das Programm enthält eine innere FOR .. NEXT-Schleife (Zeile 580 / Zeile 1030) zum Zwecke der fünfmaligen Testdurchführung für jeden Teilnehmer und eine äußere FOR .. NEXT-Schleife (Zeile 280 / Zeile 1040), deren Wiederholungshäufigkeit sich aus der Anzahl der Teilnehmer ergibt. Mittels einer abschließenden FOR .. NEXT-Schleife (Zeile 1120 / Zeile 1220) wird das Gesamtergebnis ausgegeben.

(2) Zwischen den Kommandos "AUF DIE PLAETZE!" und "FERTIG!" einerseits und zwischen "FERTIG!" und "LOS!" andererseits sind unterschiedliche, zufallsabhängige "Wartezeiten" vorgesehen, damit nicht durch einen "Gewöhnungseffekt" sehr kurze "unechte" Reaktionen gemessen werden. Die unterschiedlichen, zufallsabhängigen Wartezeiten ergeben sich durch sog. Warteschleifen in den Zeilen 640 und 670. Wenn man die Wartezeiten tendenziell verkürzen möchte, so ist die Konstante 3000 (in dem Ausdruck RND(1) * 3000) herabzusetzen (z. B. auf 2000).

(3) Die Berechnung der Reaktionsgeschwindigkeit erfolgt in den Zeilen 690 - 720. In dieser Schleifenkonstruktion wird S so lange um 1 erhöht, bis der Teilnehmer eine beliebige Taste drückt. Wenn S den Wert 0 enthält (Zeile 760), so folgt daraus, daß der Teilnehmer schon vor der Abarbeitung des Befehls in Zeile 690 eine Taste betätigt hat und S deshalb nicht um 1 erhöht wurde.

(4) Messungen haben ergeben, daß nach einer Wartezeit von 1 Minute in der Schleifenkonstruktion der Zeilen 690 - 720 die Variable S den Wert 7060 enthält. Daraus folgt, daß 1 Zählpunkt 60 / 7060 Sekunden und S Zählpunkte S * 60 / 7060 Sekunden (vgl. Zeile 960) entspricht; eine Strafsekunde erzielen wir dadurch, daß wir der Variablen S 7060 / 60 Punkte zuweisen (vgl. Zeile 870).

(5) Zum Verständnis des Programms empfehlen wir, die Wirkungsweise der unbekannten Befehle und die Ausführungen zu den indizierten Variablen (Lektion 23) zu studieren.

12. Ausgaben mit PRINT

LE 12.1

Nachdem wir in den vorangegangenen Lektionen 10 und 11 etwas über die Verarbeitung von Daten mit Hilfe der LET-Anweisung und den arithmetischen Operatoren gelernt haben, wenden wir uns zunächst der Ausgabe zu, um in den dann folgenden Lektionen die Zuweisungsbefehle READ/DATA und die Eingabeinstruktionen INPUT und GET zu behandeln.

Der Ausgabebefehl in BASIC heißt

PRINT (von engl. "to print" = drucken).

Er gehört zu den vielseitigsten Befehlen dieser Sprache.

(1) PRINT (ohne Operanden)

Die Verwendung des PRINT-Befehls ohne Operanden bewirkt die Ausgabe einer Leerzeile. Mehrere Leerzeilen erzielt man durch entsprechende Wiederholungen.

Beispiel:

```
 .
 .
40 REM AUSGABE VON 3 LEERZEILEN
50 REM -----------------------
60 PRINT
70 PRINT
80 PRINT
 .
 .
```

Abweichend von unser auf Seite 7-03 formulierten Regel, im allgemeinen nur einen Befehl pro Zeilennummer zu schreiben, können wir hier durchaus empfehlen, zur Ausgabe von drei Leerzeilen die drei PRINT-Befehle unter einer Zeilennummer zusammenzufassen, weil die Übersichtlichkeit des Programms dadurch nicht leidet, sondern eher sogar gefördert wird:

```
 .
 .
60 PRINT : PRINT : PRINT
 .
 .
```

(2) PRINT mit einem Operanden

Wenn in einem PRINT-Befehl als Operand eine Variable angegeben ist, dann wird nicht etwa der angegebene Variablenname, sondern der Inhalt des entsprechenden Speicherplatzes ausgegeben.

Beispiel:

```
10 LET A$ = "TEXT"
20 PRINT A$
RUN
TEXT
Ready   ←——— Systemmeldung nach Beendigung eines Programmlaufs
```

Wenn eine numerische Variable einen negativen Wert enthält, so wird bei ihrer Ausgabe vor der Zahl ein Minuszeichen (-) gedruckt. Bei der Ausgabe positiver Zahlen erscheint hingegen kein positives Zeichen. Statt dessen wird vor der Zahl nur eine Leerstelle freigelassen. Sowohl führende als auch nachgezogene Nullen werden unterdrückt.

Beispiel:

```
10 LET A = -16.7
20 LET B = +42
30 LET C = 004.85
40 LET D = 7.90
50 PRINT A
60 PRINT B
70 PRINT C
80 PRINT D
RUN
-16.7
 42
 4.85
 7.9
Ready
```

Die Aussagen zum Druckbild der Variablen gelten analog auch für Konstanten. (Nicht vergessen: Im Gegensatz zu numerischen Konstanten sind Anfang und Ende von Zeichenkettenkonstanten durch Anführungsstriche zu kennzeichnen!)

Beispiel:

```
10 PRINT -16.7
20 PRINT +42
30 PRINT 004.85
40 PRINT 7.90
50 PRINT "ENDE"
RUN
-16.7
 42
 4.85
 7.9
ENDE
Ready
```

(3) PRINT mit mathematischen Ausdrücken

Folgt nach einer PRINT-Anweisung ein mathematischer Ausdruck, so wird der Wert dieses Ausdrucks ausgegeben und nicht etwa dessen einzelnen Elemente.

Beispiel:

```
10 LET A = 5
20 LET B = 4
30 PRINT SQR(A↑2 + B↑2)
RUN
 6.4031242
Ready
```

PE 12.1

Mit Hilfe des Systemkommandos R U N wurde der Rechner veranlaßt, das folgende Programm zu übersetzen und auszuführen. Ergänzen Sie die fehlenden Ausgabedaten!

```
10 REM TESTAUFGABE
20 LET A1 = 7.30
30 LET A2 = 01.3
40 LET A1$ = "ZAHLEN"
50 LET A3 = -1983
60 LET A4 = +1983
70 PRINT A1$
80 PRINT
90 PRINT "A1"
100 PRINT A1
```

```
110 PRINT
120 PRINT "A2"
130 PRINT A2
140 PRINT
150 PRINT A3
160 PRINT A4
170 PRINT A1 / A2
RUN

..............................

..............................

A1
..............................

7.30
..............................

..............................

A2
..............................

1.3
..............................

..............................

-1983
..............................

1983
..............................

..............................

Ready
```

LÖS 12.1

Ausgabedaten aufgrund des BASIC-Programms aus **PE 12.1 :**

```
 .
 .
RUN
ZAHLEN
                    ◄──────── 1 Leerzeile
A1
 7.3
                    ◄──────── 1 Leerzeile
A2
 1.3
                    ◄──────── 1 Leerzeile
-1983
 1983
 5.6153846
Ready
```

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 12.1** !

LE 12.2

Will man mehrere Daten in einer Zeile ausgeben, so verwendet man zur Trennung der Operanden Kommata und Semikola.

(1) PRINT mit Formatierung durch Semikola (;)

Wenn man in einem PRINT-Befehl mehrere Operanden aufführt und diese durch Semikola trennt, so wird der folgende Operand unmittelbar nach dem vorangehenden gedruckt. (Allerdings erscheint vor positiven Zahlen eine Leerstelle für das nicht wiedergegebene Vorzeichen.)

Beispiel:

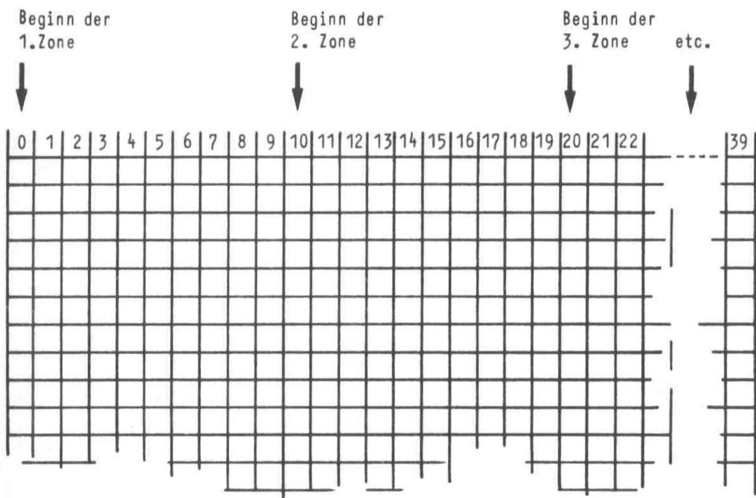
```
10 PRINT "123456789..........."
20 LET A$ = "VOR"
30 LET B$ = "NACH"
40 LET C$ = "NAME"
50 LET A1 = 7.5
60 LET A2 = -2
70 LET A3 = 6
80 PRINT
90 PRINT A$; C$
100 PRINT B$; C$
110 PRINT A1; A2; A3
RUN
123456789...........

VORNAME
NACHNAME
 7.5-2 6
Ready
```

(2) PRINT mit Formatierung durch Kommata (,)

BASIC kennt intern eine Unterteilung der Zeile in Zonen, und zwar beginnt beim SHARP MZ-700 die erste Zone mit der 1. Schreibstelle (das ist die Position 0, weil nicht ab 1, sondern ab 0 gezählt wird), die zweite Zone beginnt mit der 11. Schreibstelle (= Position 10), die dritte mit der 21. (= Position 20) und die vierte mit der 31. (= Position 30). 1)

Unterteilung des Bildschirms in Zonen:



1) Abweichend von der Aufteilung des Bildschirms sind die Anfänge der Zonen für den Plotter-Drucker (Ausgabebefehl für den Plotter-Drucker: PRINT/P) auf die Schreibpositionen 0, 8, 16, 24 und 32 festgelegt.

Wenn man nun in einer PRINT-Anweisung die Variablen (oder Konstanten oder auch mathematischen Ausdrücke) durch Kommata trennt, so erreicht man damit die Ausgabe des jeweils folgenden Wertes in derselben Zeile zu Beginn der nächsten Zone.

Die gleiche Wirkung erzielt man auch dadurch, daß man (beispielsweise) zwei Werte mit zwei PRINT-Anweisungen ausgibt und dem ersten Wert ein Komma folgen läßt.

Dabei darf man nicht übersehen, daß bei der Ausgabe von negativen Werten vor der Zahl ein Minuszeichen gedruckt wird. Hingegen erscheint bei positiven Zahlen statt des positiven Vorzeichens eine Leerstelle.

Beispiel:

```
10 PRINT "         11111111112"
20 PRINT "12345678901234567890"
30 PRINT
40 PRINT "1. ZONE:",
50 PRINT "2. ZONE:"
60 PRINT -175, -218
70 PRINT 175, 218
RUN
         11111111112
12345678901234567890

1. ZONE:  2. ZONE:
-175      -218
 175       218
Ready
```

(3) PRINT mit/ohne Zeilenvorschub

Nach jeder PRINT-Anweisung erfolgt normalerweise ein Zeilenvorschub, den das System jedoch unterdrückt, wenn dem letzten Operanden der vorangehenden PRINT-Anweisung ein Semikolon (oder Komma) folgt. Eine Unterdrückung des Zeilenvorschubs ist natürlich nur so lange möglich, wie die Kapazität einer Zeile nicht überschritten wird.

Beispiel:

```
10 PRINT "PROGRAMM";
20 PRINT "BEISPIEL"
RUN
PROGRAMMBEISPIEL
Ready
```

(4) Ausgabe von Zahlen im Gleitkommaformat

Zahlen werden als Gleitkommazahlen (vgl. hierzu auch die Erläuterungen auf Seite 8-02!) ausgegeben, wenn für den Rechner anders eine präzise Ergebnisdarstellung nicht mehr möglich ist.

Beispiel:

```
10 LET A = 123456789
20 LET B = 0.00012345678
30 PRINT A
40 PRINT B
RUN
 1.2345679E+08
 1.2345678E-04
Ready
```

(5) Ausgaben auf dem Plotter-Drucker

Wenn die Ausgaben auf dem Plotter-Drucker erfolgen sollen, so lautet das Befehlswort (statt PRINT)

PRINT/P (von engl. "to print on the printer" = auf dem Drucker drucken).

Hat der Programmierer nichts anderes bestimmt, erfolgt die Ausgabe mit s c h w a r z e n Zeichen, und zwar 40 pro Zeile.

(a) Änderung der Zeichenzahl pro Zeile

Mit folgenden Befehlen kann man die Zeichenzahl pro Zeile variieren:

MODE TN (von engl. "mode: text/normal" = Erscheinungsform: Text/normal)

schaltet den Plotter-Drucker auf Textausgabe mit 40 Zeichen pro Zeile.

MODE TL (von engl. "mode: text/large" = Erscheinungsform: Text/groß)

schaltet den Plotter-Drucker auf Textausgabe mit 26 Zeichen pro Zeile.

MODE TS (von engl. "mode: text/small" = Erscheinungsform: Text/klein)

schaltet den Plotter-Drucker auf Textausgabe mit 80 Zeichen pro Zeile.

(b) Änderung der Druckfarbe (des Plotter-Druckers)

Mit dem Befehl

PCOLOR (von engl. "printer color" = Druckerfarbe)

kann man die Farbe des Plotter-Druckers bestimmen, und zwar erzeugt

PCOLOR 0 schwarze Ausgabe,
PCOLOR 1 blaue Ausgabe,
PCOLOR 2 grüne Ausgabe und
PCOLOR 3 rote Ausgabe.

PE 12.2

Ergänzen Sie in den folgenden Beispielen die fehlenden Ausgabedaten!

(1)

```
10 LET A = 10
20 PRINT A
RUN

10..............................
Ready
```

(2)

```
10 LET B = 17.5
20 PRINT "B"
RUN

"B"..............................
Ready
```

(3)

```
10 LET A1$ = "BUCHUNGS"
20 LET A2$ = "VORGANG"
30 LET A3$ = "BELEG"
40 PRINT A1$; A2$
50 PRINT
60 PRINT A1$; A3$
RUN

buchungsvorgang..............................

..............................

buchungsbeleg..............................
Ready
```

(4)

```
10 LET A1$ = "BUCHUNGS"
20 LET A2$ = "BELEG"
30 PRINT A1$;
40 PRINT A2$
RUN

..............................
Ready
```

(5)

```
10 LET A$ = "    LISTE"
20 LET B$ = "DAMEN"
30 LET C$ = "HERREN"
40 PRINT "123456789..........."
50 PRINT A$
60 PRINT
70 PRINT B$, C$
80 PRINT 120, 210
RUN

..............................

..............................

..............................

..............................

..............................
Ready
```

LÖS 12.2

Ausgabedaten aufgrund der BASIC-Programme aus **PE 12.2** :

(1)
```
.
.
RUN
 10
Ready
```

(2)
```
.
.
RUN
B
Ready
```

(3)
```
.
.
RUN
BUCHUNGSVORGANG

BUCHUNGSBELEG
Ready
```

(4)
```
.
.
RUN
BUCHUNGSBELEG
Ready
```

(5)
```
.
.
RUN
123456789...........
    LISTE

DAMEN      HERREN
 120        210
Ready
```

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 12.1** !

ÜBUNGsprogramm zu Lektion 12

Programmbeschreibung (Eulersche Approximation):

Es gibt keinen Computer auf der Welt, bei dem nicht Rechenungenauigkeiten auftreten können. Man begegnet ihnen vor allem dann, wenn Zahlen mit sehr vielen gültigen Stellen mathematisch miteinander zu verknüpfen sind, so daß Formatgrenzen überschritten werden, und der Rechner sich dann mit Rundungen und / oder Weglassungen behelfen muß. Wann und bei welchen Operationen dies stattfindet, ist von System zu System verschieden.

Das folgende Programm dient dazu, bei BASIC-Rechnern die "Genauigkeitsgrenze" zu ermitteln. Wir bedienen uns zu diesem Zweck des Zuweisungsbefehls

LET E = (1 + 1 / Z) ↑ Z (vgl. Zeile 380).

Wenn Z eine positive Zahl enthält, dann bewirkt eine Vergrößerung von Z auch eine Vergrößerung von E; allerdings ist der "Wachstumseffekt" für E immer geringer, je größer Z wird (vgl. die folgende Aufstellung).

Z	E
1	2.000000
10	2.593742...
100	2.704813...
1000	2.7169239...
10000	2.7181459...
100000	2.7182682...
1000000	2.7182804...
10000000	2.7182818... *

* Der "wahre Wert" von e (E) für ein unendlich großes Z ist die sog. Eulersche Zahl e, die Basis der natürlichen Logarithmen.

Erhöht man nun Z innerhalb einer Schleife ständig um 1, dann resultiert daraus schließlich eine Zunahme von E in einem Nachkommastellenbereich, der jenseits der (von Rechner zu Rechner unterschiedlichen) Genauigkeitsgrenze liegt. Diese spüren wir auf durch die ständige Überprüfung, ob der jeweils vorletzte Wert von E, den wir in EA (≙ E-Alt) abgelegt haben (Zeile 340), kleiner ist als der letzte Wert von E. Wenn das nicht zutrifft, erfolgt eine entsprechende Ausgabe.

Programmliste:

```
10 REM **************************
20 REM * EULERSCHE APPROXIMATION *
30 REM **************************
40 REM
50 REM ERLAEUTERUNG DES PROGRAMMS
60 REM -------------------------
70 CLS
80 PRINT
90 PRINT "DIESES PROGRAMM ERMITTELT IN E
INER"
100 PRINT
110 PRINT "PROGRAMMSCHLEIFE EINEN NAEHER
UNGSWERT"
120 PRINT
130 PRINT "FUER DIE "; CHR$(34); "EULERS
CHE ZAHL"; CHR$(34)
140 PRINT
150 PRINT TAB(15) "e ."
160 PRINT : PRINT : PRINT
170 PRINT "DIE ARBEIT WIRD ABGEBROCHEN,
WENN DIE"
180 PRINT
190 PRINT "GENAUIGKEITSGRENZE ERREICHT I
ST."
200 FOR I = 1 TO 9 : PRINT : NEXT I
210 PRINT TAB(12) "BITTE 'W'-TASTE DRUEC
KEN!"
220 GET A$
230 IF A$ <> "W" THEN 220
240 REM
```

```
250 REM BERECHNUNG
260 REM ----------
270 CLS
280 FOR I = 1 TO 10 : PRINT : NEXT I
290 PRINT TAB(10) "BITTE WARTEN!"
300 PRINT
310 PRINT TAB(10) "ICH RECHNE!"
320 CURSOR 10, 23
330 PRINT "(VARIABLE Z =      )"
340 LET EA = E
350 LET Z = Z + 1
360 CURSOR 23, 23
370 PRINT Z;
380 LET E = (1 + 1 / Z) ↑ Z
390 IF EA < E THEN 340
400 REM
410 REM ERGEBNISAUSGABE
420 REM ---------------
430 CLS
440 PRINT : PRINT : PRINT
450 PRINT TAB(5) "GENAUIGKEITSGRENZE BEI
"
460 PRINT TAB(5) "---------------------
"
470 PRINT
480 PRINT TAB(4) Z; ". DURCHLAUF"
490 PRINT : PRINT : PRINT
500 PRINT "VORLETZTES ERGEBNIS:"; EA
510 PRINT
520 PRINT "LETZTES ERGEBNIS:   "; E
530 PRINT
540 PRINT "DIFFERENZ:          "; EA-E
550 PRINT : PRINT : PRINT
560 END
```

13. Zuweisungen mit READ und DATA

LE 13.1

Die BASIC-Befehle

READ / DATA (von engl. "to read" = lesen u. von engl. "data" = Daten)

findet man in der Literatur nicht selten unter der Überschrift "Eingabebefehle", obwohl es im strengen Sinne keine Eingabebefehle sind. [1)]

Die Anweisungen READ / DATA erfüllen vielmehr einen ähnlichen Zweck wie die LET-Anweisung. Der READ-Befehl weist die in der DATA-Anweisung angegebenen Werte den Variablen zu, die in der READ-Anweisung aufgeführt sind. Deshalb bedingen READ- und DATA-Befehl(e) einander, d. h. die Verwendung des READ-Befehls zwingt dazu, auch den DATA-Befehl einzusetzen.

Beispiel:

Die folgenden Werte sollen den angegebenen Variablen zugeordnet werden:

7 ——► A

-13 ——► B

"MAJA" ——► C$

---

1) In anderen Programmiersprachen (z. B. in COBOL) ist der gleichlautende READ-Befehl - anders als in BASIC - ein echter Eingabebefehl.

Lösung mit LET-Anweisungen:

```
10 LET A = 7
20 LET B = -13
30 LET C$ = "MAJA"
```

Lösung mit READ- und DATA-Anweisung:

```
10 READ A, B, C$
20 DATA 7, -13, "MAJA"
```

Es ist zulässig, sowohl die READ- als auch die DATA-Anweisung auf mehrere Zeilen zu verteilen. Die anschließend aufgeführten Anweisungsfolgen haben dieselbe Wirkung wie die obige.

```
10 READ A
20 READ B
30 READ C$
40 DATA 7, -13, "MAJA"
```

oder:

```
10 READ A, B, C$
20 DATA 7
30 DATA -13
40 DATA "MAJA"
```

oder:

```
10 READ A
20 READ B
30 READ C$
40 DATA 7
50 DATA -13
60 DATA "MAJA"
```

Sobald also ein READ-Befehl auftritt, wird in die dort angegebenen Variablen entsprechend der aufgeführten Reihenfolge der jeweils

nächste Wert aus der/den DATA-Anweisung(en) übertragen. Ein interner "Zeiger" weist dabei immer auf jene Konstante in der DATA-Anweisung, die bei Ausführung des nächsten READ-Befehls in einer Variablen abzulegen ist.

PE 13.1

(1) Welche der folgenden Behauptungen ist/sind richtig?

A Die BASIC-Befehle READ / DATA sind im eigentlichen Sinne keine Eingabeanweisungen. Sie erfüllen vielmehr einen ähnlichen Zweck wie die LET-Anweisung.

B Der READ-Befehl weist die in der DATA-Anweisung angegebenen Werte den Variablen zu, die in der READ-Anweisung aufgeführt sind.

C READ- und DATA-Befehl(e) bedingen einander, d. h. die Verwendung des READ-Befehls zwingt dazu, auch den DATA-Befehl aufzuführen.

D Die Befehle READ und DATA sind sog. "konträre" Befehle. Durch die Verwendung der einen Anweisung ist die Verwendung der anderen ausgeschlossen.

E Die BASIC-Anweisung READ ist ein echter Eingabebefehl, der zum Lesen von Magnetplattendateien verwendet wird.

F Den in der/den READ-Anweisung(en) angegebenen Variablen wird in der aufgeführten Reihenfolge immer der nächste Wert aus der/den DATA-Anweisung(en) zugewiesen. Ein interner "Zeiger" steht dabei immer auf jenem Wert der DATA-Anweisung, der bei Ausführung des nächsten READ-Befehls in einer Variablen abzulegen ist.

Der/die Lösungsbuchstabe/n lautet/lauten:

A, B, C, F ....

(2) Die folgenden Werte sollen den angegebenen Variablen zugewiesen werden:

218 → F1

175 → AL

"CPU" → F1$

Ergänzen Sie die Operationsteile der Befehle!

```
10 REM UEBUNGSAUFGABE
20 ............... F1
30 ............... F1$
40 ............... AL
50 ............... 218, "CPU", 175
```

(3) Die folgenden Werte sollen den angegebenen Variablen zugewiesen werden:

333 → IS

"%" → VN$

11.85 → RE

Ergänzen Sie die Operandenteile der READ-Befehle!

```
10 REM UEBUNGSAUFGABE
20 READ ............
30 READ ............
40 READ ............
50 DATA 11.85, "%", 333
```

LÖS 13.1

(1) Die Lösungsbuchstaben lauten

A, B, C, F .

(2) Die Befehle mit den ergänzten Operationsteilen haben folgendes Aussehen:

```
10 REM UEBUNGSAUFGABE
20 READ F1
30 READ F1$
40 READ AL
50 DATA 218, "CPU", 175
```

(3) Die Befehle mit den ergänzten Operandenteilen haben folgendes Aussehen:

```
10 REM UEBUNGSAUFGABE
20 READ RE
30 READ VN$
40 READ IS
50 DATA 11.85, "%", 333
```

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 13.1** !

LE 13.2

Wir kennen jetzt zwar die Wirkungsweise der BASIC-Befehle READ/DATA, doch müssen wir unser Wissen hinsichtlich einiger Details noch vervollständigen.

(1) DATA-Anweisung(en) am Programmende

Die DATA-Anweisungen können, getrennt von den READ-Befehlen, an beliebiger Stelle des Programms erscheinen. Der erfahrene Programmierer setzt aber die DATA-Anweisungen im Regelfall an das Programmende. Dadurch findet er sie - insbesondere in einem größeren Programm - zu einem späteren Zeitpunkt schnell wieder und kann ggf. die angegebenen Werte für einen neuen Programmlauf schnell ändern.

Beispiel:

```
10 REM BEISPIEL
20 REM AUTOR: HAMANN
30 READ A, B, C, D$
40 LET A = A + 1
50 LET B = B + 1.07
60 PRINT .....
  .
  .
  .
1000 DATA 7, 8, 9, "DM"
```

(2) Reihenfolge der Operanden in der/den READ- und DATA-Anweisung(en)

Die Reihenfolge der Variablentypen (numerische Variablen - String-Variablen) in der/den READ-Anweisung(en) muß mit der Reihenfolge der entsprechenden Werte in der/den DATA-An-

weisung(en) übereinstimmen; insbesondere darf einer numerischen Variablen keine Zeichenkettenkonstante zugewiesen werden. Bei Nichtbeachtung dieser Regel unterbricht das System den Programmlauf, und folgende Fehlermeldung erscheint:

`Illegal data error in ...` (= Fehler wegen unzulässiger Daten in Zeile ...)

Beispiel:

```
10 READ F, A$, B
 .
 .
 .
1000 DATA "DM", 200, 50
RUN
Illegal data error in 10
Ready
```

Erläuterungen:

Sofort nachdem das Systemkommando R U N erteilt wurde, beendete der Rechner die Arbeit mit der Fehlermeldung Illegal data error in 10, weil der numerischen Variablen F die Zeichenkettenkonstante "DM" nicht zugewiesen werden darf. Zusätzlich erscheint die Systemmeldung R E A D Y (= fertig).

(3) Anzahl der Variablen im READ-Befehl ungleich der Anzahl der Werte im DATA-Befehl

Wenn das System feststellt, daß die Anzahl der Variablen in den READ-Befehlen größer ist als die in den DATA-Anweisungen aufgeführten Werte, dann wird der Programmlauf mit folgender Fehlermeldung unterbrochen:

`READ error in ...` (= READ-Fehler in Zeile ...)

Beispiel:

```
10 READ A, B
20 READ C$
  .
  .
  .
1000 DATA 100, 200
RUN
READ error in 20
Ready
```

Erläuterungen:

Nachdem der Variablen A der Wert 100 und der Variablen B der Wert 200 zugewiesen worden ist, steht für die String-Variable C$ keine Zeichenkettenkonstante in der DATA-Anweisung zur Verfügung, so daß die Fehlermeldung READ error in 20 erscheint.

Wenn hingegen die Anzahl der Variablen in der/den READ-Anweisung(en) geringer ist als die Anzahl der in der/den DATA-Anweisung(en) aufgeführten Werte, nimmt das System daran keinen Anstoß.

Beispiel:

```
10 READ A
20 DATA 100, 200
RUN
Ready
```

(4) Unzulässigkeit von mathematischen Ausdrücken im DATA-Befehl

In einer DATA-Anweisung dürfen nur die in Lektion 8 erläuterten Konstantentypen erscheinen, nicht hingegen mathematische Ausdrücke wie z. B. 7/5 oder SQR(A * 5).

(5) Darstellung von Zeichenkettenkonstanten in der DATA-Anweisung

Es ist nicht zwingend, in der DATA-Anweisung eine Zeichenkettenkonstante durch Anführungszeichen zu kennzeichnen.

Beispiel:

```
10 READ A$
.
.
.
100 DATA "LOGIK DER PROGRAMMIERUNG"
```

auch zulässig:

```
10 READ A$
.
.
.
100 DATA LOGIK DER PROGRAMMIERUNG
```

(6) READ / DATA im Verhältnis zu LET

Einem Anfänger fehlt bisweilen die Einsicht dafür, weshalb es in BASIC neben dem Zuweisungsbefehl LET noch die Anweisungen READ / DATA gibt. Der Vorzug von READ / DATA gegenüber LET sei an einem Beispiel erläutert:

Innerhalb eines größeren Programms wurden an vielen verschiedenen Stellen numerische Werte den vorgesehenen Variablen zugewiesen. Nach einem Programmlauf besteht die Absicht, eine weitere Berechnung durchzuführen, allerdings mit anderen (zuzuweisenden) Werten. Bei Verwendung des LET-Befehls muß das Programm durchsucht und an vielen verschiedenen Stellen geändert werden, was sehr mühselig sein kann. Bei Benutzung von READ / DATA hingegen ist (sind) nur der (die) DATA-Befehl(e) auszutauschen, der (die) sinnvollerweise am Programmende steht (stehen) und daher leicht zu finden ist (sind).

PE 13.2

(Bitte, die linke Seite abdecken!)

In jedem der folgenden Programmausschnitte ist ein Fehler enthalten, der von Ihnen gekennzeichnet werden soll.

(1)
```
10 READ A, B, E
 .
 .
100 DATA 7, 20.9, "DM"
```

Beschreibung des Fehlers: ....

....

(2)
```
10 READ A, FE, SU, B$
 .
 .
100 DATA 10, 15, 18
```

Beschreibung des Fehlers: ....

....

(3)
```
10 READ P
20 LET D = P
 .
 .
100 DATA 8↑2
```

Beschreibung des Fehlers: ....

....

LÖS 13.2

(1)
```
10 READ A, B, E
 .
 .
100 DATA 7, 20.9, "DM"
```

Beschreibung des Fehlers: Es liegt ein "Illegal data error" vor! Der numerischen Variablen E darf eine Zeichenkettenkonstante - hier: "DM" - nicht zugewiesen werden. (Fehlermeldung: Illegal data error in 10)

(2)
```
10 READ A, FE, SU, B$
 .
 .
100 DATA 10, 15, 18
```

Beschreibung des Fehlers: Für die String-Variable B$ steht in der DATA-Anweisung keine Zeichenkettenkonstante zur Verfügung! (Fehlermeldung: READ error in 10)

(3)
```
10 READ P
20 LET D = P
 .
 .
100 DATA 8↑2
```

Beschreibung des Fehlers: In der DATA-Anweisung dürfen keine mathematischen Ausdrücke erscheinen! (Fehlermeldung: Illegal data error in 10)

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur ➞ **LE 13.2** !

ÜBUNGsprogramm zu Lektion 13

Programmbeschreibung (Buchstabenmemory):

Eine vom Bediener einzugebende Menge von Buchstaben soll zufallsabhängig ausgewählt und auf dem Bildschirm ausgegeben werden. Nach einer bestimmten Zeit, die umso länger dauert, je höher die Anzahl der Buchstaben ist, wird die Zeichenkette gelöscht. Der Spieler hat anschließend jene Zeichen einzugeben, an die er sich erinnern kann, jedoch dürfen maximal so viele Buchstaben genannt werden, wie es zu Anfang festgelegt worden ist. - Anschließend ermittelt das Programm die "Anzahl der Treffer" und die "Treffer in Prozent".

Tritt in der vom System angegebenen Buchstabenfolge ein Buchstabe mehrfach auf, so genügt es, wenn der Anwender diesen einmal aufführt. Auch die Reihenfolge der Nennungen unterliegt nicht der Bewertung.

Programmliste:

```
10 REM ********************
20 REM * BUCHSTABENMEMORY *
30 REM ********************
40 REM
50 REM BILD-1
60 REM ------
70 CLS
80 FOR I = 1 TO 10 : PRINT : NEXT I
90 PRINT TAB(9) "********************"
100 PRINT TAB(9) "*                  *"
110 PRINT TAB(9) "* BUCHSTABENMEMORY *"
120 PRINT TAB(9) "*                  *"
130 PRINT TAB(9) "********************"
140 FOR I = 1 TO 3000 : NEXT I
150 REM
```

```
160 REM ZUWEISUNG DER BUCHSTABEN MIT
170 REM READ ZUR STRING-VARIABLEN A$
180 REM ----------------------------
190 READ A$
200 REM
210 REM BILD-2
220 REM ------
230 CLS
240 FOR I = 1 TO 10 : PRINT : NEXT I
250 PRINT TAB(5) "AUS WIEVIEL BUCHSTABEN
 SOLL"
260 PRINT
270 PRINT TAB(5) "DIE ZU BILDENDE ZEICHE
NKETTE"
280 PRINT
290 PRINT TAB(5) "BESTEHEN ";
300 INPUT Z
310 IF Z = 0 THEN 1040
320 REM
330 REM BILDUNG DER ZEICHENKETTE
340 REM ------------------------
350 REM
360 LET B$ = ""
370 FOR I = 1 TO Z
380 LET X = INT(26 * RND(1) + 1)
390 LET B$ = B$ + MID$(A$, X, 1)
400 NEXT I
410 REM
420 REM BILD-3
430 REM ------
440 REM (AUSGABE DER ZEICHENKETTE FUER
450 REM EINEN KURZEN ZEITRAUM)
460 CLS
470 FOR I = 1 TO 10 : PRINT : NEXT I
480 PRINT TAB(5) B$
490 FOR I = 1 TO (500 * Z): NEXT I
500 REM
510 REM BILD-4
520 REM ------
530 CLS
540 FOR I = 1 TO 3 : PRINT : NEXT I
550 PRINT TAB(5) "NENNEN SIE DIE BUCHSTA
BEN, DIE SIE"
560 PRINT
570 PRINT TAB(5) "ERKANNT HABEN?"
```

```
580 PRINT
590 PRINT TAB(5) "(MAXIMAL"; Z; " BUCHST
ABEN, DIE"
600 PRINT
610 PRINT TAB(5) "SIE OHNE TRENNUNGSSTRI

CHE ETC."
620 PRINT
630 PRINT TAB(5) "AUFFUEHREN SOLLEN?"
640 PRINT : PRINT
650 INPUT "      → "; C$
660 REM UEBERPRUEFUNG DER EINGABE
670 REM ------------------------
680 IF LEN(C$) <= Z THEN 790
690 CLS
700 FOR I = 1 TO 10 : PRINT : NEXT I
710 PRINT TAB(5) "EINGABEFEHLER?"
720 PRINT
730 PRINT TAB(5) "DAS WAREN ZU VIELE BUC
HSTABEN?"
740 PRINT
750 PRINT TAB(5) "EINGABE WIEDERHOLEN?"
760 FOR I = 1 TO 5000 : NEXT I
770 GOTO 510
780 REM
790 REM AUSWERTUNG DER EINGABE
800 REM ----------------------
810 REM
820 LET T = 0
830 FOR I = 1 TO Z
840 FOR K = 1 TO LEN(C$)
850 IF MID$(C$, K, 1) = MID$(B$, I, 1) T
HEN 880
860 NEXT K
870 GOTO 890
880 LET T = T + 1
890 NEXT I
900 REM
910 REM BILD-5
920 REM ------
930 CLS
940 FOR I = 1 TO 10 : PRINT : NEXT I
950 PRINT TAB(5) "ANZAHL DER TREFFER:
 "; T
```

```
960 PRINT
970 PRINT TAB(5) "TREFFERZAHL IN PROZENT
: ";
980 PRINT USING "###.##"; 100 * T / Z
990 FOR I = 1 TO 8 : PRINT : NEXT I
1000 PRINT TAB(12) "BITTE 'W'-TASTE DRUE
CKEN!"
1010 GET W$
1020 IF W$ <> "W" THEN 1010
1030 GOTO 210
1040 END
1050 DATA "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

14. Zurücksetzen des DATA-Zeigers mit RESTORE

LE 14

Wir wollen zunächst im 1. Teil eines Programmbeispiels noch einmal die Anwendung der Befehle READ/DATA zeigen. In dem auf dem 1. Teil aufbauenden 2. Teil werden wir die RESTORE-Anweisung erklären.

Programmbeispiel

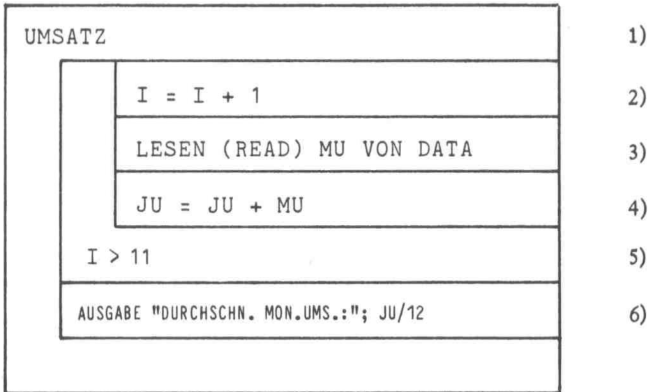
(1) Aufgabendefinition (1. Teil)

Aus den 12 Monatsumsätzen einer Unternehmung ist der durchschnittliche Monatsumsatz zu berechnen. (Die Monatsumsätze sind nacheinander mit dem READ-Befehl der Variablen MU zuzuweisen und mit Hilfe der Variablen JU zu addieren. Die Division von JU durch 12 ergibt den durchschnittlichen Monatsumsatz. Das Ergebnis ist auszugeben.)

Aufgabendefinition (2. Teil)

Anschließend soll ermittelt werden, wieviel Prozent die vier Quartalsumsätze vom Gesamtumsatz ausmachen.

(2) Reihenfolgeplanung (Struktogramm) / 1. Teil



Zuzuweisende Daten:

99485, 76356, 78999, 45384,
66666, 77333, 48956, 55432,
86123, 98765, 56734, 34575

---

1) Der Name des Programms lautet "UMSATZ".

2) Die Variable I verwenden wir zum Zählen der Schleifendurchläufe.

3) Die Monatsumsätze werden innerhalb der Schleife nacheinander der Variablen MU zugewiesen.

4) Mit Hilfe der Variablen JU addieren wir die Monatsumsätze. (Zu Beginn des Programmlaufs erhalten die numerischen Variablen bei einem BASIC-System den Anfangswert 0.)

5) Nachdem die Befehle der Schleife 12mal ausgeführt worden sind - dann hat I einen Wert größer als 11 - ist die Z-Schleife beendet.

6) Das Ergebnis wird ausgegeben.

(3) Befehle zum nebenstehenden Struktogramm (incl. Verarbeitungsergebnis)

```
10 REM * UMSATZ *
20 LET I = I + 1
30 READ MU
40 LET JU = JU + MU
50 IF I > 11 THEN 70
60 GOTO 20
70 PRINT "DURCHSCHN. MON.UMS.:"; JU/12
80 DATA 99485, 76356, 78999
90 DATA 45384, 66666, 77333
100 DATA 48956, 55432, 86123
110 DATA 98765, 56734, 34575
RUN
DURCHSCHN. MON.UMS.: 68734
Ready
```

Wir wollen uns jetzt dem 2. Teil der Aufgabe zuwenden.

Nach der Ausgabe des durchschnittlichen Monatsumsatzes sind die Werte der DATA-Anweisung erneut mit dem READ-Befehl der Variablen MU zuzuweisen, damit sowohl die Quartalsumsätze als auch deren Prozentanteil vom Gesamtumsatz errechnet werden können.

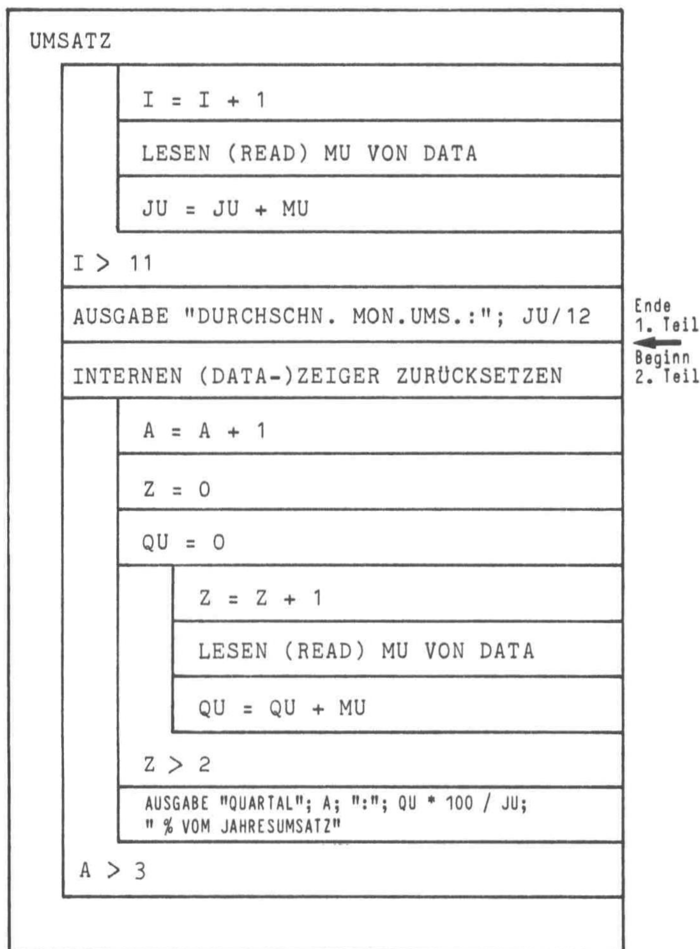
Daraus folgt, daß der interne "Zeiger", der grundsätzlich auf jenen Wert des DATA-Befehls verweist, der bei der Ausführung des nächsten READ-Befehls in einer Variablen abzulegen ist, "zurückgestellt" werden muß. Hierfür verwenden wir die Anweisung

RESTORE (von engl. "to restore" = zurückstellen).

Sie wird immer dann eingesetzt, wenn mit denselben Daten mehr als eine Berechnung durchzuführen ist.

Ergänzender Hinweis: Wenn nach der RESTORE-Anweisung eine Zeilennummer erscheint, wird der interne "Zeiger" auf den ersten Wert der damit gekennzeichneten DATA-Anweisung gestellt.

(4) Reihenfolgeplanung (Struktogramm) / 1. und 2. Teil



Zuzuweisende Daten:

99485, 76356, 78999, 45384, 66666, 77333,
48956, 55432, 86123, 98765, 56734, 34575

Zu Beginn des 2. Teils setzen wir den internen "(DATA-)Zeiger" mit RESTORE zurück.

Jeweils drei Monatsumsätze werden in der inneren Schleife mit der Variablen QU addiert, wobei wir die Variable Z zum Zählen der drei Schleifendurchläufe verwenden. Die Variable A dient einerseits bei der Ergebnisausgabe zur Kennzeichnung des jeweiligen Quartals und andererseits als Zähler für die äußere Schleife. (Damit wir die Variablen Z und QU mehrmals für die Berechnung in der inneren Schleife verwenden können, müssen wir beide Datenfelder auf 0 setzen.)

(5) Befehle zum nebenstehenden Struktogramm (incl. Verarbeitungsergebnisse)

```
10 REM * UMSATZ *
20 LET I = I + 1
30 READ MU
40 LET JU = JU + MU
50 IF I > 11 THEN 70
60 GOTO 20
70 PRINT "DURCHSCHN. MON.UMS.:"; JU/12
80 RESTORE
90 LET A = A + 1
100 LET Z = 0
110 LET QU = 0
120 LET Z = Z + 1
130 READ MU
140 LET QU = QU + MU
150 IF Z > 2 THEN 170
160 GOTO 120
170 PRINT "QUARTAL"; A; ":"; QU * 100 /
JU; " % VOM JAHRESUMSATZ"
180 IF A > 3 THEN 200
190 GOTO 90
200 END
210 DATA 99485, 76356, 78999, 45384
220 DATA 66666, 77333, 48956, 55432
230 DATA 86123, 98765, 56734, 34575
```

```
RUN
DURCHSCHN. MON.UMS.: 68734
QUARTAL 1: 30.896888 % VOM JAHRESUMSATZ
QUARTAL 2: 22.960859 % VOM JAHRESUMSATZ
QUARTAL 3: 23.097618 % VOM JAHRESUMSATZ
QUARTAL 4: 23.044636 % VOM JAHRESUMSATZ
Ready
```

PE 14

Welche der folgenden Behauptungen ist/sind richtig?

A Der RESTORE-Befehl weist die in der DATA-Anweisung angegebenen Werte den Variablen zu, die in der RESTORE-Anweisung aufgeführt sind.

B Der interne "(DATA-)Zeiger", der grundsätzlich auf jenen Wert des DATA-Befehls weist, der bei der Ausführung des nächsten READ-Befehls in einer Variablen abzulegen ist, kann mit Hilfe des Befehls RESTORE "zurückgestellt" werden.

C Nach der Durchführung des RESTORE-Befehls weist der interne "(DATA-)Zeiger" auf den ersten Wert der DATA-Anweisung(en).

D Der Befehl RESTORE wird immer dann eingesetzt, wenn mit denselben Daten mehr als eine Berechnung durchzuführen ist.

E Wenn nach der RESTORE-Anweisung eine Zeilennummer erscheint, wird der interne "(DATA-)Zeiger" auf den ersten Wert der damit gekennzeichneten DATA-Anweisung gestellt.

F RESTORE ist ein Systemkommando, durch das der Rechner veranlaßt wird, das im Arbeitsspeicher befindliche Programm zu übersetzen und auszuführen.

Der/die Lösungsbuchstabe/n lautet/lauten

....

LÖS 14

Die Lösungsbuchstaben lauten

B, C, D, E .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 13.1** !

ÜBUNGsprogramm zu Lektion 14

Problemstellung:

Zwölf Personen unterhalten sich über eine Zahl, zwei von ihnen lügen, die anderen sagen die Wahrheit.

1. Person: "Die Zahl ist ein Vielfaches von 2."
2. Person: "Die Zahl ist ein Vielfaches von 3."
3. Person: "Die Zahl ist ein Vielfaches von 5."
4. Person: "Die Zahl ist ein Vielfaches von 7."
5. Person: "Die Zahl enthält u. a. eine Ziffer 7."
6. Person: "Die Zahl enthält u. a. eine Ziffer 3."
7. Person: "Die Zahl enthält u. a. eine Ziffer 0."
8. Person: "Die Summe der einzelnen Ziffern ist größer als 9." (Beispiel: Die Summe der Ziffern der Zahl 375 wäre 15, denn 3 + 7 + 5 = 15.)
9. Person: "Die Summe der einzelnen Ziffern ist durch 5 teilbar."
10. Person: "Die Zahl ist kleiner als 873."
11. Person: "Die Zahl ist kleiner als 433."
12. Person: "Die Zahl ist größer als 333."

Programmbeschreibung (Ermittlung einer unbekannten Zahl):

Im folgenden Programm erhöhen wir die Variable Z, mit 0 beginnend, immer wieder um 1 und überprüfen dann jede so gewonnene Zahl hinsichtlich der genannten Bedingungen, und zwar so lange, bis ein Wert gefunden ist, für den 10 Bedingungen zutreffen. (Durch Modifikation des Programms lassen sich sehr leicht andere Zahlen finden, die ohne Rechner wesentlich schwieriger zu ermitteln sind.)

Programmliste:

```
10 REM ****************
20 REM * ZAHLENRAETSEL *
30 REM ****************
40 CLS
50 DIM P(12)
60 REM
70 REM ERLAEUTERUNG DER VARIABLEN:
80 REM --------------------------
90 REM
100 REM (1) REGISTRIERUNG DER WAHREN
110 REM     AUSSAGEN (BEDINGUNGEN) MIT
120 REM     DEN INDIZIERTEN VARIABLEN
130 REM     P(X)
140 REM
150 REM (2) ZAEHLEN DER ANZAHL DER
160 REM     RICHTIGEN BEDINGUNGEN MIT
170 REM     DER VARIABLEN
180 REM     S
190 REM
200 REM (3) DIE JEWEILS UNTERSUCHTE
210 REM     ZAHL IST IN DER VARIABLEN
220 REM     Z
230 REM
240 REM LOESCHEN DER ARBEITSVARIABLEN
250 REM P(X) UND S / W5, W6, W7, B8
260 REM BEGINN DER UNTERSUCHUNG
270 REM =============================
280 LET S = 0
290 FOR I = 1 TO 12
300 LET P(I) = 0
310 NEXT I
320 LET W5=0 : W6=0 : W7=0 : B8=0
330 LET Z = Z + 1
340 REM AUSGABE DER UNTERSUCHTEN ZAHL
350 REM -----------------------------
360 CURSOR 30,2
370 PRINT Z;
380 REM BEDINGUNG-1:
390 REM -----------
400 IF (Z/2) <> INT(Z/2) THEN 420
410 LET P(1) = 1 : LET S = S + 1
```

```
420 REM BEDINGUNG-2:
430 REM -----------
440 IF (Z/3) <> INT(Z/3) THEN 460
450 LET P(2) = 1 : LET S = S + 1
460 REM BEDINGUNG-3:
470 REM -----------
480 IF (Z/5) <> INT(Z/5) THEN 500
490 LET P(3) = 1 : LET S = S + 1
500 REM BEDINGUNG-4:
510 REM -----------
520 IF (Z/7) <> INT(Z/7) THEN 540
530 LET P(4) = 1 : LET S = S + 1
540 REM BEDINGUNG-5:
550 REM -----------
560 LET Z$ = STR$(Z)
570 LET L = LEN(Z$)
580 FOR I = 1 TO L
590 IF MID$(Z$,I,1) = "7" THEN W5 = 1
600 NEXT I
610 IF W5 = 0 THEN 630
620 LET P(5) = 1 : LET S = S + 1
630 REM BEDINGUNG-6:
640 REM -----------
650 FOR I = 1 TO L
660 IF MID$(Z$,I,1) = "3" THEN W6 = 1
670 NEXT I
680 IF W6 = 0 THEN 700
690 LET P(6) = 1 : LET S = S + 1
700 REM BEDINGUNG-7:
710 REM -----------
720 FOR I = 1 TO L
730 IF MID$(Z$,I,1) = "0" THEN W7 = 1
740 NEXT I
750 IF W7 = 0 THEN 770
760 LET P(7) = 1 : LET S = S + 1
770 REM BEDINGUNG-8:
780 REM -----------
790 FOR I = 1 TO L
800 LET B8$ = MID$(Z$,I,1)
810 LET B8 = B8 + VAL(B8$)
820 NEXT I
830 IF B8 > 9 THEN P(8)=1 : S=S+1
```

```
840 REM BEDINGUNG-9:
850 REM ------------
860 IF (B8 / 5) <> INT(B8 / 5) THEN 880
870 LET P(9) = 1 : LET S = S + 1
880 REM BEDINGUNG-10:
890 REM -------------
900 IF Z < 873 THEN P(10)=1 : S=S+1
910 REM BEDINGUNG-11:
920 REM -------------
930 IF Z < 433 THEN P(11)=1 : S=S+1
940 REM BEDINGUNG-12:
950 REM -------------
960 IF Z > 333 THEN P(12)=1 : S=S+1
970 REM
980 REM ENDEBEDINGUNG:
990 REM =============
1000 IF S < 10 THEN 240
1010 REM
1020 REM ERGEBNISAUSGABE:
1030 REM ----------------
1040 CLS
1050 PRINT
1060 PRINT S; " BEDINGUNGEN SIND RICHTIG
§"
1070 PRINT
1080 FOR I = 1 TO 12
1090 IF P(I) = 1 THEN PRINT " PERSON"; I
; " SAGT DIE WAHRHEIT§"
1100 NEXT I
1110 PRINT
1120 PRINT " DIE ZAHL LAUTET"; Z
1130 END
```

15. Eingaben mit INPUT und GET

LE 15.1

Um Daten während der Programmausführung eingeben zu können, gibt es in BASIC den Befehl

INPUT (von engl. "input" = Eingabe).

Bei der Ausführung der Anweisung druckt der Rechner ein Fragezeichen (?) und wartet dann, bis der Benutzer einen entsprechenden Wert eingegeben und die [CR]-Taste gedrückt hat. Das System speichert den Wert in jener Variablen, die nach dem Operationsteil (INPUT) aufgeführt ist.

Beispiel:

```
10 REM BEISPIEL
20 PRINT "BITTE ZAHL EINGEBEN!"
30 INPUT A
40 PRINT "WURZEL VON"; A; " :"; SQR(A)
50 END
RUN
BITTE ZAHL EINGEBEN!
?
```

Aufgrund des INPUT-Befehls hat der Rechner das Fragezeichen ausgegeben und wartet nun, bis eine Eingabe erfolgt.

```
.
.
? 17.3
WURZEL VON 17.3 : 4.1593269
Ready
```

(17.3 ⟶ vom Benutzer eingegeben)

Es ist zumeist ratsam, dem INPUT-Befehl eine PRINT-Anweisung vorangehen zu lassen, durch die der Benutzer darüber informiert wird,

was er zu tun hat, wenn aufgrund der INPUT-Anweisung ein Fragezeichen erscheint.

Statt des der INPUT-Anweisung vorangestellten PRINT-Befehls ist es auch möglich, direkt mit dem INPUT-Befehl eine Nachricht auszugeben (Das System generiert dann aber kein Fragezeichen!):

```
10 REM BEISPIEL
20 INPUT "RADIUS (IN M) EINGEBEN! "; RA
30 LET FL = π * RA↑2
40 PRINT "KREISFLAECHE (IN QM):"
50 PRINT FL
60 END
RUN
RADIUS (IN M) EINGEBEN! 17        (17 → vom Benutzer eingegeben)
KREISFLAECHE (IN QM):
 907.92028
Ready
```

Es ist zulässig, nach dem Operationsteil (INPUT) mehrere Variablen - sowohl numerische als auch String-Variablen - aufzuführen, die durch Kommata voneinander zu trennen sind. Die Anzahl und Art der dann einzugebenden Werte muß den im INPUT-Befehl aufgeführten Variablen entsprechen:

```
10 REM BEISPIEL
20 INPUT "3 ZAHLEN EINGEBEN! "; A, B, C
30 PRINT "ARITHMETISCHER MITTELWERT:"
40 PRINT (A + B + C) / 3
50 END
RUN
3 ZAHLEN EINGEBEN! 16.07, 19, 14.33
ARITHMETISCHER MITTELWERT:
 16.466667
Ready
```

Ergänzende Hinweise:

(1) Einzugebende Strings müssen nicht in Anführungszeichen eingeschlossen sein.

(2) Bei der Eingabe eines unzulässigen Wertes (String statt einer Zahl) unterbricht das System die Verarbeitung mit der Fehler-

meldung "Illegal data error in ..".

Beispiel:

```
10 REM BEISPIEL
20 INPUT "RADIUS EINGEBEN! "; RA
30 PRINT "KREISFLAECHE:"
40 PRINT π * RA↑2
RUN
RADIUS EINGEBEN! FUENF   ←— unzulässige Eingabe des Anwenders
Illegal data error in 20
Ready
```

(3) Wenn der Benutzer zu wenig Werte eingegeben hat, dann druckt das System so lange ein Fragezeichen, bis alle Variablen versorgt sind.

Beispiel:

```
10 REM BEISPIEL
20 PRINT "3 ZAHLEN EINGEBEN!"
30 INPUT A, B, C
40 PRINT "SUMME DER DREI ZAHLEN:"
50 PRINT A + B + C
60 END
RUN
3 ZAHLEN EINGEBEN!
? 87
? 886.6
? 433.333
SUMME DER DREI ZAHLEN:
 1406.933
Ready
```

(4) Wenn der Anwender mehr Werte eingibt, als Variablen im INPUT-Befehl angegeben sind, werden die zuviel angegebenen Zahlen bzw. Strings nicht berücksichtigt.

(5) Es ist auch zulässig, mit Hilfe von INPUT indizierten Variablen Werte zuzuweisen. (Allerdings müssen die indizierten Variablen - anders als bei vielen anderen BASIC-Dialekten - in jedem Fall vorher mit DIM eingerichtet worden sein; vgl. hierzu Lektion 23!)

Programmbeispiel

Aufgabendefinition

Der Rechner soll ermitteln, auf welchen Betrag ein beliebiges (einzugebendes) Anfangskapital mit Zins und Zinseszins in einer beliebigen (einzugebenden) Anzahl von Jahren anwächst. Hierbei ist die übliche Berechnungsmethode - also jene mit jährlichem Zuschlag der Zinsen - zu verwenden.

Der Zinssatz ist nach Festlegungszeiträumen gestaffelt. Der Programmlauf soll beendet werden, wenn ein Anfangskapital von 0 DM eingegeben wird.

Folgende Variablennamen sind zu verwenden:

AK = AnfangsKapital

FJ = Festlegungsdauer in Jahren

Z = Zinssatz
(Zinssätze:
für FJ kleiner/gleich 4 gilt Z = 5;
für FJ größer 4 aber kleiner/gleich 8 gilt Z = 7;
für FJ größer 8 gilt Z = 9.)

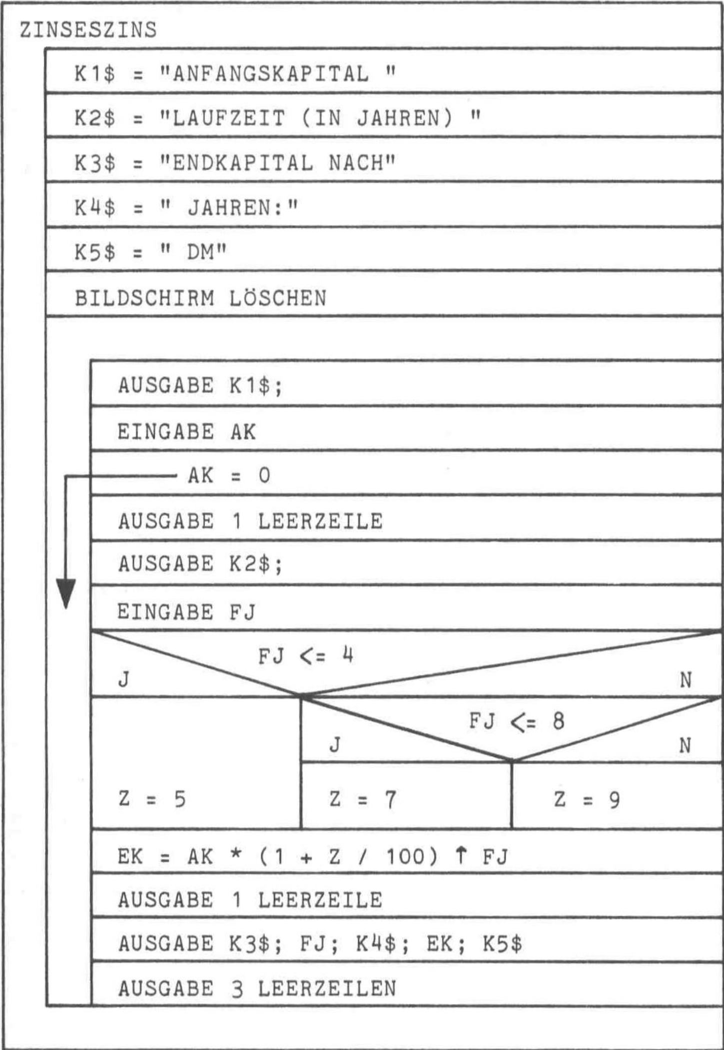
EK = EndKapital

Das Endkapital läßt sich mit der folgenden Formel (sog. Zinseszinsformel) berechnen:

EK = AK * (1 + Z / 100) ↑ FJ

EK → (Endkapital), AK → (Anfangskapital), Z → (Zinssatz), FJ → (Festlegungsdauer in Jahren)

Reihenfolgeplanung (Struktogramm)



PE 15.1

Ergänzen Sie im folgenden BASIC-Programm zum nebenstehenden Struktogramm die fehlenden Befehle!

```
10 REM **************
20 REM * ZINSESZINS *
30 REM **************
40 LET K1$ = "ANFANGSKAPITAL "
50 LET K2$ = "LAUFZEIT (IN JAHREN) "
60 LET K3$ = "ENDKAPITAL NACH"
70 LET K4$ = " JAHREN:"
80 LET K5$ = " DM"

90 CLS ....................................
100 REM BEGINN DER M-SCHLEIFE
110 REM --------------------
120 PRINT K1$;

130 INPUT AK ..............................
140 IF AK = 0 THEN 320 : REM -----
150 PRINT

160 PRINT K2$ .............................

170 INPUT FJ ..............................
180 IF FJ <= 4 THEN 240
190 IF FJ <= 8 THEN 220
200 LET Z = 9
210 GOTO 250
220 LET Z = 7
230 GOTO 250
240 LET Z = 5
250 LET EK = AK * (1 + Z / 100) ↑ FJ
260 PRINT

270 .......................................
280 PRINT : PRINT : PRINT
290 GOTO 100
300 REM ENDE DER M-SCHLEIFE
310 REM ------------------
320 END
```

LÖS 15.1

BASIC-Programm der **PE 15.1** mit den ergänzten Befehlen:

```
10 REM **************
20 REM * ZINSESZINS *
30 REM **************
40 LET K1$ = "ANFANGSKAPITAL "
50 LET K2$ = "LAUFZEIT (IN JAHREN) "
60 LET K3$ = "ENDKAPITAL NACH"
70 LET K4$ = " JAHREN:"
80 LET K5$ = " DM"
90 CLS
100 REM BEGINN DER M-SCHLEIFE
110 REM ---------------------
120 PRINT K1$;
130 INPUT AK
140 IF AK = 0 THEN 320 : REM -----
150 PRINT
160 PRINT K2$
170 INPUT FJ
180 IF FJ <= 4 THEN 240
190 IF FJ <= 8 THEN 220
200 LET Z = 9
210 GOTO 250
220 LET Z = 7
230 GOTO 250
240 LET Z = 5
250 LET EK = AK * (1 + Z / 100) ↑ FJ
260 PRINT
270 PRINT K3$; FJ; K4$; EK; K5$
280 PRINT : PRINT : PRINT
290 GOTO 100
300 REM ENDE DER M-SCHLEIFE
310 REM -------------------
320 END
```

Sollten Ihre Ergänzungen nicht richtig sein, überlegen Sie selbst, welche **LE** Sie wiederholen müssen!

LE 15.2

Neben der INPUT-Anweisung gibt es noch den Befehl

GET (von engl. "to get" = holen),

der auch für Eingabezwecke verwendet wird.

Zur Erinnerung: Mit INPUT kann man Zahlen oder Strings eingeben. Solange die Eingabe, die aus mehreren Zeichen bestehen darf, nicht abgeschlossen ist, wartet das System. Geschriebene Zeichen lassen sich (u. a.) durch Betätigung der [DEL]-Taste wieder löschen, denn erst nach dem Drücken der [CR]-Taste ist die Eingabe abgeschlossen, und der Programmlauf wird fortgesetzt.

Demgegenüber sind die Eigenschaften der GET-Anweisung völlig anders:

(1) Die Eingabe darf nur aus einem einzigen Zeichen bestehen.

(2) Das eingegebene Zeichen wird nicht auf dem Bildschirm protokolliert.

(3) Die Eingabe wird nicht durch Betätigung der [CR]-Taste abgeschlossen.

(4) Der Rechner druckt kein Fragezeichen und wartet nicht, bis die Eingabe eines Zeichens ausgeführt worden ist.

Das für den Anfänger schwierigste Problem besteht darin, daß das System bei GET nicht auf den Vollzug der Eingabe wartet.

Programmiert man beispielsweise

```
.
.
200 PRINT "BELIEBIGE ZIFFER EINGEBEN!"
210 GET X
.
.
```

dann stehen bei Ausführung von Zeile 210 ein paar Millisekunden Zeit zur Verfügung, in denen das System ein Zeichen einlesen und der Variablen X zuweisen kann. Drückt der Benutzer zu spät auf die Taste, dann hat der Rechner den Wert nicht speichern können, und die Variable X behält ihren ursprünglichen Wert (ggf. den Anfangswert 0).

Der GET-Befehl ist daher nur sinnvoll, wenn er wiederholt in einer Schleife ausgeführt wird, z. B. folgendermaßen:

```
.
.
200 PRINT "ZIFFER ZWISCHEN 1 - 9 EINGEBEN!"
210 GET X
220 IF X = 0 THEN 210
.
.
```

Solange X den Wert 0 hat - sei es der Anfangswert der Variablen, sei es durch Drücken der 0-Taste - verzweigt das Programm in Befehlszeile 220 immer wieder zurück zum GET in Zeile 210. Die Schleife wird erst verlassen, wenn man eine andere Ziffer als 0 eingegeben hat.

Wozu benötigt man die GET-Anweisung?

Man benötigt sie vor allem, um verschiedenartige Warteschleifen zu programmieren, die erst nach Betätigung einer bestimmten Taste verlassen werden.

Beispiele:

```
10 CLS
20 PRINT "BILD 1"
30 FOR I = 1 TO 20 : PRINT : NEXT I
40 PRINT "W-TASTE DRUECKEN!"
50 GET A$
60 IF A$ <> "W" THEN 50
70 CLS
80 PRINT "BILD 2"
90 PRINT : PRINT : PRINT
100 END
```

Erläuterungen:

Zeile 10: Bildschirm wird gelöscht

Zeile 20: Ausgabe des Strings "BILD 1"

Zeile 30: Ausgabe von 20 Leerzeilen (vgl. hierzu Lektion 20)

Zeile 40: Ausgabe des Strings "W-TASTE DRUECKEN!"

```
BILD 1

                    (20 Leerzeilen)

W-TASTE DRUECKEN!
```

Zeile 50 u.

Zeile 60: Warteschleife, die so lange durchlaufen wird, wie die String-Variable A$ ungleich (<>) "W" ist.

Anders ausgedrückt: Erst nach dem Drücken der "W"-Taste wird die Programmausführung in Zeile 70 fortgesetzt.

Zeile 70: Bildschirm wird gelöscht

Zeile 80: Ausgabe des Strings "BILD 2"

Zeile 90: Ausgabe von 3 Leerzeilen

Zeile 100: Programmende

```
BILD 2

                    (3 Leerzeilen)
Ready
```

Wenn man möchte, daß die Warteschleife bei Betätigung einer beliebigen Taste verlassen wird, dann müßte in Zeile 60 abgefragt werden, ob die String-Variable A$ eine "leere Zeichenkette" enthält. Eine leere Zeichenkette besteht aus null Zeichen, d. h. aus zwei Anführungszeichen "mit" nichts dazwischen: "" .
Die Befehle 40 - 60 würden also folgendermaßen aussehen:

```
 .
 .
40 PRINT "BELIEBIGE TASTE DRUECKEN!"
50 GET A$
60 IF A$ = "" THEN 50
 .
 .
```

(2)

```
 .
 .
50 PRINT "WIEDERHOLUNG? (J/N)"
60 GET A$
70 IF A$ = "J" THEN 10
80 IF A$ = "N" THEN 100
90 GOTO 60
 .
 .
```

Erläuterungen:

Die Warteschleife der Zeile 60 bis 90 wird nur verlassen, wenn der Anwender entweder (für "JA") die "J"-Taste (dann Verzweigung nach Zeile 10) oder (für "NEIN") die "N"-Taste drückt (dann Verzweigung nach Zeile 100).

(3)

```
    .
    .
1000 PRINT "WELCHE BERECHNUNGSART?"
1010 PRINT "1, 2, 3 oder 4?"
1020 GET A
1030 ON A GOTO 2000, 3000, 4000, 5000
1040 GOTO 1020
    .
    .
```

Erläuterungen:

Die obige Warteschleife ermöglicht eine Verzweigung an vier verschiedene Stellen des Programms nach Eingabe der Ziffer 1 oder 2 oder 3 oder 4. Der Anfänger sollte das Beispiel zunächst übergehen, da wir erst in Lektion 22 die Verzweigung mit ON .. GOTO .. behandeln werden.

Ergänzender Hinweis:

Als Sonderzubehör gibt es für den SHARP MZ-700 eine Interface-Karte, über die zusätzliche externe Geräte angeschlossen und dann mit den folgenden Anweisungen angesprochen werden können:

INP #P, D (liest 8-Bit-Daten vom Port P, wandelt sie in Dezimalwerte um und weist sie der Variablen D zu.)

OUT #P, D (wandelt eine Dezimalzahl, die in der Variablen D vorliegt, in einen Binärwert um und gibt ihn an Port P aus. D darf zwischen 0 und 255 liegen.)

(Vgl. hierzu ggf. die Unterlagen des Herstellers!)

PE 15.2

(1) Welche der folgenden Behauptungen ist/sind richtig?

A Mit dem GET-Befehl kann man nur ein einziges Zeichen eingeben.

B Das eingegebene Zeichen wird beim GET nicht auf dem Bildschirm protokolliert.

C Der Rechner druckt beim GET kein Fragezeichen und wartet nicht, bis die Eingabe eines Zeichens ausgeführt worden ist.

D Jede Eingabe mit der GET-Anweisung muß durch Betätigung der CR-Taste abgeschlossen werden.

E Man benötigt GET vor allem, um verschiedenartige Warteschleifen zu programmieren, die erst nach Betätigung einer bestimmten Taste verlassen werden.

Der/die Lösungsbuchstabe/n lautet/lauten

A, B, d, E ....

(2) Im folgenden Programm sollen die zu ergänzenden Befehlszeilen 50 und 60 eine Warteschleife bilden, deren Ausführung das System unterbricht, wenn der Anwender die "A"-Taste drückt.

```
10 CLS
20 PRINT "TEIL 1"
30 FOR I = 1 TO 20 : PRINT : NEXT I
40 PRINT "A-TASTE DRUECKEN!"

50 GET A$..........................

60 IF A$="" THEN 50................
70 CLS
80 PRINT "TEIL 2"
90 END
```

LÖS 15.2

(1) Die Lösungsbuchstaben lauten

A, B, C, E .

(2) BASIC-Programm mit den ergänzten Befehlszeilen 50 und 60 (Warteschleife):

```
10 CLS
20 PRINT "TEIL 1"
30 FOR I = 1 TO 20 : PRINT : NEXT I
40 PRINT "A-TASTE DRUECKEN!"
50 GET A$
60 IF A$ <> "A" THEN 50
70 CLS
80 PRINT "TEIL 2"
90 END
```

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 15.2** !

ÜBUNGsprogramm zu Lektion 15

Programmbeschreibung (Würfelspiel):

Das folgende Programm simuliert ein Würfelspiel für beliebig viele Personen. Zwei Würfel befinden sich im Spiel. Wer an der Reihe ist, kann so häufig würfeln, wie er möchte. Die Augenzahl wird immer addiert und dem jeweiligen Konto gutgeschrieben, wenn der Teilnehmer

- die Würfel weiterreicht
- und keine '6' gewürfelt hat.

Wenn hingegen beim Würfeln eine '6' erscheint, ist der nächste Spieler an der Reihe, und die Punkte werden nicht verbucht.

Wer zuerst mindestens 200 Punkte erreicht hat, gilt als Sieger.

Programmliste:

```
10 REM ***************
20 REM * WUERFELSPIEL *
30 REM ***************
40 GOSUB 1000 : REM BILD-1
50 GOSUB 2000 : REM BILD-2
60 GOSUB 3000 : REM BILD-3
70 GOSUB 4000 : REM BILD-4
80 GOSUB 5000 : REM BILD-5
90 END
1000 REM BILD-1:
1010 REM =======
1020 CLS
1030 PRINT
1040 FOR I = 1 TO 22
1050 PRINT " *"
1060 NEXT I
```

```
1070 PRINT TAB(1);
1080 FOR I = 1 TO 37
1090 PRINT "*";
1100 NEXT I
1110 FOR I = 23 TO 1 STEP -1
1120 CURSOR 38, I
1130 PRINT "*"
1140 NEXT I
1150 FOR I = 38 TO 2 STEP -1
1160 CURSOR I, 1
1170 PRINT "*"
1180 NEXT I
1190 CURSOR  8, 12
1200 FOR I = 1 TO 12
1210 LET UE$ = MID$("WUERFELSPIEL",I,1)
1220 PRINT UE$; " ";
1230 FOR K = 1 TO 1000 : NEXT K
1240 NEXT I
1250 RETURN
2000 REM BILD-2 (SPIELREGELN):
2010 REM ====================
2020 CLS
2030 PRINT
2040 PRINT " S P I E L R E G E L N :"
2050 PRINT " ----------------------
2060 PRINT
2070 PRINT " FUER DAS SPIEL WERDEN ZWEI
WUERFEL"
2080 PRINT " VERWENDET. JEDER TEILNEHMER
 DARF DAMIT"
2090 PRINT " BELIEBIG HAEUFIG WUERFELN.
DIE AUGEN-"
2100 PRINT " ZAHL WIRD ADDIERT UND SEINE
M KONTO"
2110 PRINT " GUTGESCHRIEBEN, WENN ER
2120 PRINT
2130 PRINT " - DIE WUERFEL WEITERREICHT"
2140 PRINT
2150 PRINT " - UND KEINE '6' GEWUERFELT
HAT."
2160 PRINT
2170 PRINT " WENN HINGEGEN BEIM WUERFELN
 EINE '6' "
```

```
2180 PRINT " ERSCHEINT, IST DER NAECHSTE
 SPIELER"
2190 PRINT " AN DER REIHE, UND DIE PUNKT
E WERDEN"
2200 PRINT " NICHT VERBUCHT."
2210 PRINT
2220 PRINT " WER ZUERST 200 PUNKTE ERREI
CHT HAT,"
2230 PRINT " GILT ALS SIEGER!"
2240 PRINT : PRINT
2250 PRINT TAB(20) "'W'-TASTE DRUECKEN!"
2260 GET A$ : IF A$ <> "W" THEN 2260
2270 RETURN
3000 REM BILD-3 (TEILNEHMERZAHL)
3010 REM ======================
3020 CLS
3030 FOR I = 1 TO 10 : PRINT : NEXT I
3040 PRINT TAB(5) "ANZAHL DER SPIELER ";
3050 INPUT SZ
3060 DIM GP(SZ)
3070 RETURN
4000 REM BILD-4 (WUERFELN)
4010 REM ================
4020 CONSOLE 0, 25, 0, 40
4030 CLS
4040 IF SZ = Z THEN LET Z = 0
4050 LET Z = Z + 1
4060 LET RS = 0
4070 CONSOLE 5, 20
4080 PRINT
4090 PRINT TAB(12) "SPIELER NR."; Z; ":"
4100 PRINT " ---------------------------
---------- "
4110 CLS
4120 PRINT : PRINT : PRINT
4130 PRINT TAB(10) "(DER WUERFELBECHER"
4140 PRINT TAB(10) "WIRD GESCHUETTELT!)"
4150 FOR I = 1 TO 10 : PRINT : NEXT I
4160 PRINT TAB(11) "WUERFELN MIT DER"
4170 PRINT
4180 PRINT TAB(14) "'W'-TASTE!"
4190 GET A$
4200 LET W = RND(1)
4210 IF A$ <> "W" THEN 4190
```

```
4220 CLS
4230 PRINT : PRINT : PRINT
4240 LET W1 = INT(RND(1) * 6 + 1)
4250 LET W2 = INT(RND(1) * 6 + 1)
4260 PRINT " ERREICHTE AUGENZAHL:"; W1;
" +"; W2
4270 PRINT : PRINT
4280 IF (W1=6) + (W2=6) THEN 4510
4290 LET RS = RS + W1 + W2
4300 PRINT " SUMME DER PUNKTE"
4310 PRINT " IN DIESER RUNDE:  "; RS
4320 PRINT : PRINT
4330 PRINT " WEITERWUERFELN ? (J/N)"
4340 GET A$
4350 IF A$ = "J" THEN 4110
4360 IF A$ = "N" THEN 4380
4370 GOTO 4340
4380 CLS
4390 LET GP(Z) = GP(Z) + RS
4400 PRINT : PRINT : PRINT
4410 PRINT " IHRE GESAMTPUNKTZAHL"
4420 PRINT " (KONTOSTAND) :"
4430 PRINT
4440 PRINT TAB(10) GP(Z)
4450 IF GP(Z) > 199 THEN 4650
4460 FOR I = 1 TO 10 : PRINT : NEXT I
4470 PRINT TAB(15)"'W'-TASTE DRUECKEN!"
4480 GET A$
4490 IF A$ <> "W" THEN 4480
4500 GOTO 4000
4510 REM PUNKTE-ANNULLIERUNG
4520 REM ------------------
4530 PRINT " IHNEN WERDEN FUER DIESE RUN
DE KEINE"
4540 PRINT
4550 PRINT " PUNKTE GUTGESCHRIEBEN, DA S
IE EINE"
4560 PRINT
4570 PRINT TAB(10) "'6'"
4580 PRINT
4590 PRINT " GEWUERFELT HABEN!"
4600 PRINT : PRINT : PRINT
```

```
4610 PRINT TAB(9) "'W'-TASTE DRUECKEN!"
4620 GET A$
4630 IF A$ <> "W" THEN 4620
4640 GOTO 4000
4650 RETURN
5000 REM BILD-5 (SIEGEREHRUNG)
5010 REM ====================
5020 PRINT
5030 PRINT " SIE SIND DER SIEGER!
5040 PRINT
5050 PRINT " PUNKTSTAND SAEMTLICHER SPIE
LER:"
5060 PRINT " ---------------------------
----"
5070 PRINT
5080 FOR I = 1 TO SZ
5090 PRINT " SPIELER NR."; I; ": ";
5100 PRINT USING "###"; GP(I);
5110 PRINT " PUNKTE"
5120 NEXT I
5130 CONSOLE 0, 25, 0, 40
5140 RETURN
```

16. Farbe und Grafik

LE 16.1

Wie kaum ein anderer Rechner bietet der MZ-700 dem Anwender eine Vielzahl von unterschiedlichen Zeichen, Grafik-Befehlen und farblichen Gestaltungsmöglichkeiten, und zwar sowohl für den Bildschirm als auch für den Plotter-Drucker.

Wenden wir uns zunächst dem Zeichensatz zu:

(1) Großbuchstaben / Ziffern / Sonderzeichen

Von den Funktionstasten (z. B. [CR], [SHIFT], [INST], [DEL]) abgesehen, stehen uns durch Drücken nur der jeweiligen Taste jene Zeichen zur Verfügung, die auf den Berührungsflächen stehen. Das sind vor allem

- Großbuchstaben,
- Ziffern und
- Sonderzeichen.

(2) Tasten mit zwei Zeichen auf der Berührungsfläche

Hat eine Taste auf der Berührungsfläche zwei Zeichen - wie beispielsweise [=/-] - dann erhält man das untere Zeichen (-) durch das Drücken nur dieser Taste und das obere Zeichen (=) durch g l e i c h z e i t i g e s Drücken einer [SHIFT]-Taste (sie ist wie bei der Schreibmaschine zweimal vorhanden) und der jeweiligen Zeichentaste. (Betätigt man eine Buchstabentaste zusammen mit [SHIFT] , so erhält man Kleinbuchstaben.)

(3) Grafische Zeichen und Symbole

Alle grauen Tasten haben auf der Stirnseite weitere grafische Zeichen und Symbole; z. B. ist die [A]-Taste auf der linken Stirnseite mit [♥] belegt und auf der rechten Stirnseite mit [♣].

Um die grafischen Zeichen der linken Stirnseite zu schreiben, muß man vorher die [GRAPH]-Taste einmal betätigt haben. (Das Cursor-Muster ändert sich von "░" zu "▚".)

Die grafischen Zeichen der rechten Stirnseite erhält man (nachdem die [GRAPH]-Taste gedrückt worden ist) durch gleichzeitige Betätigung der [SHIFT]-Taste und der Zeichentaste.

Will man statt der grafischen Zeichen wieder die Zeichen der Berührungsflächen - also beispielsweise Buchstaben - schreiben, so erzielt man die "Umschaltung" mit Hilfe der [ALPHA]-Taste. (Das Cursor-Muster hat dann wieder sein ursprüngliches Aussehen: ░.)

(4) Groß- und Kleinbuchstaben

Nachdem man gleichzeitig [SHIFT] und [ALPHA] gedrückt hat, reagiert die Tastatur des MZ-700 ähnlich der einer Schreibmaschine. Es stehen dann zur Verfügung

- die Zeichen der Berührungsflächen als Kleinbuchstaben (bei den Buchstabentasten),
- die oberen Zeichen der Berührungsflächen (wenn diese zwei Zeichen enthalten),
- Großbuchstaben, wenn man die Buchstabentasten gleichzeitig mit [SHIFT] betätigt,
- die unteren Zeichen der Berührungsflächen (wenn diese zwei Zeichen enthalten) durch gleichzeitiges Drücken von [SHIFT] und der gewählten Taste(n).

Durch nochmaliges simultanes Drücken von [SHIFT] und [ALPHA] bewirken wir die Wiederherstellung des Normalzustands.

(5) Bestimmung der Farbkombination

Die standardmäßige Farbkombination des Bildschirms besteht aus

- der Farbe Weiß für die Zeichen und
- der Farbe Blau für den Hintergrund.

Für alle unter Pkt. (1) bis (5) aufgeführten Zeichen lassen sich andere Farbkombinationen bestimmen, und zwar stehen sowohl für die Zeichen als auch für den Hintergrund je 8 verschiedene Farben zur Verfügung, nämlich

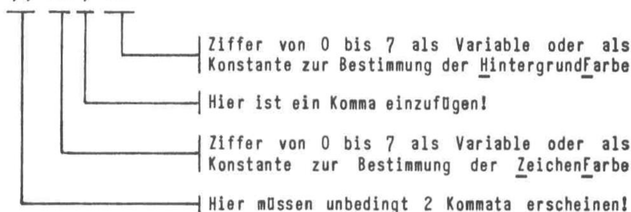
Farbe	Farbnummer
schwarz	0
blau	1
rot	2
purpurrot	3
grün	4
hellblau	5
gelb	6
weiß	7

Das Befehlswort zum Umschalten der Farbkombination lautet

COLOR (von am. engl. "color" = Farbe).

Um eine konkrete Farbkombination zu erzielen, ist das Befehlswort COLOR um die folgenden Angaben zu ergänzen:

COLOR ,, ZF, HF



Beispiele:

COLOR ,, 0, 1	erzeugt für alle weiteren Ausgaben die Zeichenfarbe Schwarz (0) und die Hintergrundfarbe Blau (1)
COLOR ,, 2, 0	erzeugt für alle weiteren Ausgaben die Zeichenfarbe Rot (2) und die Hintergrundfarbe Schwarz (0)
COLOR ,, 4, 2	erzeugt für alle weiteren Ausgaben die Zeichenfarbe Grün (4) und die Hintergrundfarbe Rot (2)
COLOR ,,, 0	ändert für alle weiteren Ausgaben nur die Hintergrundfarbe in Schwarz
COLOR ,, 1	ändert für alle weiteren Ausgaben nur die Zeichenfarbe in Blau
COLOR ,, 7, 1	stellt für alle weiteren Ausgaben die Standardeinstellung, nämlich Zeichenfarbe Weiß und Hintergrundfarbe Blau, wieder her

Mit Hilfe des folgenden Programms kann man sich eine Übersicht sämtlicher Möglichkeiten verschaffen. Nach dem Drücken der [W]-Taste erscheint die nächste Farbkombination. (Nicht alle Kombinationen sind für Wiedergabezwecke gleich günstig. Schon bei der 1. Kombination - Zeichenfarbe 0 und Hintergrundfarbe 0 - sind naturgemäß keine Zeichen zu erkennen!)

Programm:

```
10 REM ************
20 REM * FARBTEST *
30 REM ************
40 FOR ZF = 0 TO 7
50 FOR HF = 0 TO 7
60 COLOR ,, ZF, HF
70 CLS
80 PRINT "WIEDERGABE"
```

```
90 PRINT
100 PRINT "DER FARBKOMBINATION"
110 PRINT : PRINT
120 PRINT "ZEICHENFARBE NR.:"; ZF
130 PRINT
140 PRINT "    MIT"
150 PRINT
160 PRINT "HINTERGRUNDFARBE NR.:"; HF
170 GET A$
180 IF A$ <> "W" THEN 170
190 NEXT HF
200 NEXT ZF
210 COLOR ,, 7, 1
220 CLS
230 END
```

Hinweis: Das Programm hat in erster Linie die Aufgabe, alle Farbkombinationen aufzuzeigen und die jeweiligen Werte für die Zeichen- und die Hintergrundfarbe zu benennen. Für das Verständnis der Funktionsweise sind vor allem Kenntnisse über Schleifenkonstruktionen mit FOR ... TO .. / NEXT .. erforderlich (vgl. hierzu Lektion 20).

PE 16.1

(1) Welches Zeichen erzielt man durch Betätigung der Taste [A]?

A.........................

(2) Welches Zeichen erzielt man durch gleichzeitige Betätigung von [SHIFT] und [A]?

a.........................

(3) Welches Zeichen erzielt man, nachdem die [GRAPH]-Taste einmal gedrückt worden ist, durch Betätigung der [A]-Taste?

....

(4) Welches Zeichen erzielt man, nachdem die [GRAPH]-Taste einmal gedrückt worden ist, durch gleichzeitige Betätigung von [SHIFT] und [A].

....

(5) Welches Zeichen erzielt man, nachdem [SHIFT] zusammen mit [ALPHA] einmal gedrückt worden ist, durch Betätigung der [A]-Taste?

a.........................

(6) Welches Zeichen erzielt man, nachdem [SHIFT] zusammen mit [ALPHA] einmal gedrückt worden ist, durch gleichzeitige Betätigung von [SHIFT] und [A]?

A ....

(7) Wodurch wird der ursprüngliche Zustand wieder hergestellt, wenn man [SHIFT] zusammen mit [ALPHA] einmal gedrückt hat?

[illegible] ....

(8) Purpurrot hat die Farbnummer 3 und Schwarz hat die Farbnummer 0. – Wie lautet der Befehl, mit dessen Hilfe man für alle folgenden Ausgaben die Zeichenfarbe Schwarz und die Hintergrundfarbe Purpurrot erzeugen kann?

COLOR,,0,3 ....

(9) Grün hat die Farbnummer 4. Wie lautet der Befehl, mit dem man für alle folgenden Ausgaben (nur) die Zeichenfarbe in Grün ändert?

COLOR,,4 ....

(10) Wie lautet der Befehl, mit dem man für alle folgenden Ausgaben (nur) die Hintergrundfarbe in Grün ändert?

COLOR,,,4 ....

LÖS 16.1

(1) Welches Zeichen erzielt man durch Betätigung der Taste [A]?

Antwort: (den Großbuchstaben) A

(2) Welches Zeichen erzielt man durch gleichzeitige Betätigung von [SHIFT] und [A]?

Antwort: (den Kleinbuchstaben) a

(3) Welches Zeichen erzielt man, nachdem die [GRAPH]-Taste einmal gedrückt worden ist, durch Betätigung der [A]-Taste?

Antwort: (das linke Zeichen der Stirnseite:) ♥

(4) Welches Zeichen erzielt man, nachdem die [GRAPH]-Taste einmal gedrückt worden ist, durch gleichzeitige Betätigung von [SHIFT] und [A].

Antwort: (das rechte Zeichen der Stirnseite:) ♣

(5) Welches Zeichen erzielt man, nachdem [SHIFT] zusammen mit [ALPHA] einmal gedrückt worden ist, durch Betätigung der [A]-Taste?

Antwort: (den Kleinbuchstaben) a

(6) Welches Zeichen erzielt man, nachdem [SHIFT] zusammen mit [ALPHA] einmal gedrückt worden ist, durch gleichzeitige Betätigung von [SHIFT] und [A]?

Antwort: (den Großbuchstaben) A

(7) Wodurch wird der ursprüngliche Zustand wieder hergestellt, wenn man [SHIFT] zusammen mit [ALPHA] einmal gedrückt hat?

Antwort: durch nochmalige Betätigung von [SHIFT] und [ALPHA]

(8) Purpurrot hat die Farbnummer 3 und Schwarz hat die Farbnummer 0. - Wie lautet der Befehl, mit dessen Hilfe man für alle folgenden Ausgaben die Zeichenfarbe Schwarz und die Hintergrundfarbe Purpurrot erzeugen kann?

Antwort: COLOR ,, 0, 3

(9) Grün hat die Farbnummer 4. Wie lautet der Befehl, mit dem man für alle folgenden Ausgaben (nur) die Zeichenfarbe in Grün ändert?

Antwort: COLOR ,, 4

(10) Wie lautet der Befehl, mit dem man für alle folgenden Ausgaben (nur) die Hintergrundfarbe in Grün ändert?

Antwort: COLOR ,,, 4

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 16.1** !

LE 16.2

In der vorangegangenen **LE** haben wir gelernt, daß man mit der Anweisung

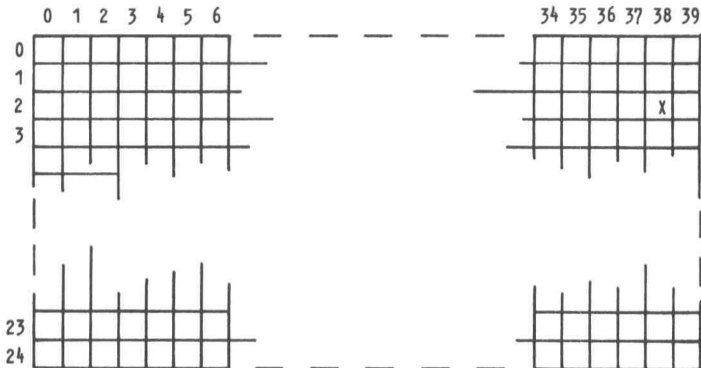
COLOR ,, ZF, HF

eine neue Kombination von ZeichenFarbe und HintergrundFarbe für alle folgenden Ausgaben auf den Bildschirm festlegen kann.

Darüber hinaus ist es möglich, mit diesem Befehl für j e d e e i n z e l n e S c h r e i b s t e l l e des Bildschirms eine betimmte Zeichen- und Hintergrundfarbe festzulegen.

Wir erinnern uns an die Darstellung auf Seite 9-07:

Der Bildschirm ist in 40 Spalten (Spalte 0 - Spalte 39) und 25 Zeilen (Zeile 0 - Zeile 24) unterteilt:



Jede Zeichenstelle des Bildschirms läßt sich mit zwei Angaben genau lokalisieren, nämlich mit der

- Nennung der Spalte und der

- Nennung der Zeile.

Die Zeichenstelle, die in der Darstellung auf Seite 16-10 ein "X" enthält, hat also

die Position 38, 2
(Spalte) (Zeile)

Wenn wir nun nach dem Befehlswort COLOR vor dem ersten Komma einen Wert zwischen 0 und 39 (für die Spalte) und vor dem zweiten Komma einen Wert zwischen 0 und 24 (für die Zeile) - sei es als Variable, sei es als Konstante - angeben, so teilen wir dem System dadurch mit, daß die folgenden Angaben für die ZeichenFarbe und die HintergrundFarbe nur für die spezifizierte Position gelten sollen.

Daraus ergibt sich das allgemeine Format der COLOR-Anweisung:

```
COLOR S, Z, ZF, HF
```

- HF: HintergrundFarbe (0 - 7)
- ZF: ZeichenFarbe (0 - 7)
- Z: Zeile (0 - 24)
- S: Spalte (0 - 39)

Die Anwendung des obigen Befehls wollen wir an einem kleinen Programm verdeutlichen, das lediglich dazu dient, ein Gedicht von Heinz Erhardt auf dem Bildschirm wiederzugeben.

Das erste Bildschirmbild zeigt folgenden Text, wobei die Überschrift aufgrund der Befehlszeilen 100 - 120 mit roter Zeichenfarbe und schwarzer Hintergrundfarbe wiedergegeben wird:

```
          Der Koenig Erl

    (Frei nach Johann Wolfgang
    von Frankfurt)



          von Heinz Erhardt
```

Nach Betätigung einer beliebigen Taste erscheinen die folgenden Verszeilen, deren jeweiliger Anfangsbuchstabe durch die Befehle der Zeilen 270, 310, 340, 380, 420, 450, 490 und 530 ebenfalls durch rote Zeichenfarbe und schwarze Hintergrundfarbe hervorgehoben ist:

```
Wer reitet so spaet
  durch Wind und Nacht?

Es ist der Vater.
  Es ist gleich acht.

Im Arm den Knaben er wohl haelt,

er haelt ihn warm,
  denn er ist erkaelt'.

Halb drei, halb fuenf.
  Es wird schon hell.

Noch immer reitet der Vater schnell.

Erreicht den Hof
  mit Mueh und Not ----

der Knabe lebt,
  das Pferd ist tot!
```

Nach abermaliger Betätigung einer beliebigen Taste wird die Systemmeldung Ready ausgegeben.

Programmliste:

```
10 REM ***********
20 REM * PROGERL *
30 REM ***********
40 REM
50 REM UEBERSCHRIFT
60 REM ------------
70 CLS
80 PRINT : PRINT : PRINT
90 PRINT TAB(10) "Der Koenig Erl"
100 FOR S = 10 TO 23
110 COLOR S, 3, 2, 0
120 NEXT S
130 PRINT
140 PRINT "     (Frei nach Johann Wolfgan
g"
150 PRINT "     von Frankfurt)"
160 PRINT : PRINT : PRINT
170 PRINT TAB(10) "von Heinz Erhardt"
180 GET A$
190 IF A$ = "" THEN 180
200 REM
210 REM VERSE
220 REM -----
230 CLS
240 PRINT
250 PRINT "Wer reitet so spaet"
260 PRINT "  durch Wind und Nacht?"
270 COLOR 0, 1, 2, 0
280 PRINT
290 PRINT "Es ist der Vater."
300 PRINT "  Es ist gleich acht."
310 COLOR 0, 4, 2, 0
320 PRINT
330 PRINT "Im Arm den Knaben er wohl hae
lt,"
340 COLOR 0, 7, 2, 0
350 PRINT
```

```
360 PRINT "er haelt ihn warm,"
370 PRINT "  denn er ist erkaelt'."
380 COLOR 0, 9, 2, 0
390 PRINT
400 PRINT "Halb drei, halb fuenf."
410 PRINT "  Es wird schon hell."
420 COLOR 0, 12, 2, 0
430 PRINT
440 PRINT "Noch immer reitet der Vater s
chnell."
450 COLOR 0, 15, 2, 0
460 PRINT
470 PRINT "Erreicht den Hof"
480 PRINT "  mit Mueh und Not ----"
490 COLOR 0, 17, 2, 0
500 PRINT
510 PRINT "der Knabe lebt,"
520 PRINT "  das Pferd ist tot!"
530 COLOR 0, 20, 2, 0
540 GET A$
550 IF A$ = "" THEN 540
560 CLS
570 END
```

PE 16.2

Welche der folgenden Behauptungen ist/sind richtig?

A Das allgemeine Format des Befehls, mit dem man für alle weiteren Ausgaben auf dem Bildschirm die Kombination der Zeichen- und Hintergrundfarbe ändern kann, lautet

COLOR ,, ZF, HF .

B Das allgemeine Format des Befehls, mit dem man für jede spezifizierte Zeichenstelle des Bildschirms eine bestimmte Kombination von Zeichen- und Hintergrundfarbe festlegen kann, lautet

COLOR S, Z, ZF, HF .

C Jede Zeichenstelle des Bildschirms läßt sich mit Nennung der Spalte (0 - 39) und der Zeile (0 - 24) genau lokalisieren.

D Eine Variierung der Zeichen- und Hintergrundfarbe des Bildschirms ist grundsätzlich nicht möglich, weil damit eine Überbeanspruchung des Systems verbunden wäre.

E Für die Festlegung der Zeichen- und Hintergrundfarbe stehen je 8 Werte (0 - 7) zur Verfügung, nämlich 0 = schwarz, 1 = blau, 2 = rot, 3 = purpurrot, 4 = grün, 5 = hellblau, 6 = gelb und 7 = weiß.

Der/die Lösungsbuchstabe/n lautet/lauten

A, B, C, E ....

LÖS 16.2

Die Lösungsbuchstaben lauten

A, B, C, E

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 16.1** !

LE 16.3

In der vorangegangenen **LE 16.1** und **LE 16.2** haben wir gelernt, daß man mit Hilfe der COLOR-Anweisung

- entweder generell für alle weiteren Ausgaben
- oder speziell für genau festgelegte Positionen

die Kombination von Zeichen- und Hintergrundfarbe ändern kann.

Darüber hinaus ist es möglich, durch die zusätzliche Angabe einer "Farbspezifikation" zum PRINT-Befehl für alle mit dieser Anweisung auszugebenden Daten eine gesonderte Farbkombination festzulegen:

```
PRINT [ZF, HF] ...
```

... : auszugebende Variablen oder Konstanten

] : eckige rechte Klammer

(Achtung: nicht verwechseln mit der sonst üblichen rechten Klammer, wie sie z. B. für mathematische Ausdrücke zu verwenden ist!)

ZF, HF : Farbspezifikation

(Sie gilt nur für die Operanden der PRINT-Anweisung. - Die Spezifikation kann sowohl mit Variablen als auch mit Konstanten vorgenommen werden. - Darüber hinaus ist es möglich, mit nur einer Angabe entweder die Zeichenfarbe:

```
PRINT [ZF,] ...
```

oder die Hintergrundfarbe:

```
PRINT [, HF] ...
```

zu verändern.)

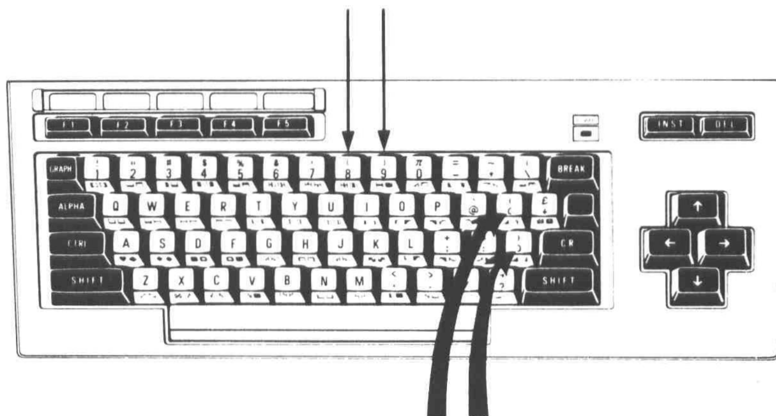
ZF = ZeichenFarbe
HF = HintergrundFarbe

[: eckige linke Klammer

(Achtung: nicht verwechseln mit der sonst üblichen linken Klammer, wie sie z. B. für mathematische Ausdrücke zu verwenden ist!)

Klammern für die Farbspezifikation beim PRINT-Befehl:

runde Klammern (z. B. für mathematische Ausdrücke)



eckige Klammern für die Farbspezifikation

Das folgende Testprogramm bietet die Möglichkeit, die Wirkungsweise des "PRINT mit Farbspezifikation" zu testen.

Programmliste:

```
10 REM FARBKOMBINATION MIT PRINT
20 REM (TESTPROGRAMM)
30 REM -------------------------
40 CLS
```

```
50 PRINT "ZEICHENFARBE (0-7)";
60 INPUT ZF
70 IF (ZF < 0) + (ZF > 7) THEN PRINT "EI          *
NGABEFEHLER!" : GOTO 50
80 PRINT
90 PRINT "HINTERGRUNDFARBE (0-7)";
100 INPUT HF
110 IF (HF < 0) + (HF > 7) THEN PRINT "E          *
INGABEFEHLER!" : GOTO 90
120 CLS
130 PRINT [ZF, HF] "KOMB.:"; ZF; ","; HF
140 PRINT : PRINT : PRINT
150 PRINT "WEITERE VERSUCHE?"
160 GET JN$
170 IF JN$ = "J" THEN 40
180 IF JN$ = "N" THEN 200
190 GOTO 160
200 END
```

* In der hier vorgestellten BASIC-Version weichen die Befehle in den Zeilen 70 und 110 von der sonst üblichen Schreibweise ab. Etwas verbreiteter ist die folgende Form (die beim SHARP MZ-700 aber nicht zulässig ist):

```
70 IF ZF < 0 OR ZF > 7 THEN PRINT ...
```

bzw.

```
110 IF HF < 0 OR HF > 7 THEN PRINT ...
```

(Vgl. hierzu Lektion 18!)

PE 16.3

Wir wollen unser **ÜBUNG**sprogramm zu Lektion 9 ("UHRZEIT3") wieder aufgreifen. Die Befehle der Zeilen 200 - 400 (vgl. Seite 9-08), nämlich

```
.
.
200 PRINT "LOS ANGELES"
210 PRINT
220 PRINT "CHICAGO"
230 PRINT
240 PRINT "NEW YORK"
250 PRINT
260 PRINT "RIO DE JANEIRO"
270 PRINT
280 PRINT "DAKAR"
290 PRINT
300 PRINT "** HAMBURG **"
310 PRINT
320 PRINT "KAPSTADT"
330 PRINT
340 PRINT "MOSKAU"
350 PRINT
360 PRINT "HONGKONG"
370 PRINT
380 PRINT "TOKIO"
390 PRINT
400 PRINT "SYDNEY"
.
.
```

sind unter Verwendung der auf Seite 16-03 angegebenen Werte für die Farben dergestalt zu ändern, daß die Orte

LOS ANGELES, CHICAGO, NEW YORK, RIO DE JANEIRO, DAKAR mit roter Zeichenfarbe und schwarzer Hintergrundfarbe,

** HAMBURG **
mit weißer Zeichenfarbe und schwarzer Hintergrundfarbe,

KAPSTADT, MOSKAU, HONGKONG, TOKIO, SYDNEY
mit grüner Zeichenfarbe und roter Hintergrundfarbe

wiedergegeben werden.

Vervollständigen Sie unter Berücksichtigung der obigen Angaben den folgenden Programmausschnitt!

```
    .
    .

200 ..............................
210 PRINT

220 ..............................
230 PRINT

240 ..............................
250 PRINT

260 ..............................
270 PRINT

280 ..............................
290 PRINT

300 ..............................
310 PRINT

320 ..............................
330 PRINT

340 ..............................
```

```
350 PRINT

360 .............................
370 PRINT

380 .............................
390 PRINT

400 .............................
 .
 .
```

LÖS 16.3

Der vervollständigte Programmausschnitt hat folgendes Aussehen:

```
.
.
200 PRINT [2, 0] "LOS ANGELES"
210 PRINT
220 PRINT [2, 0] "CHICAGO"
230 PRINT
240 PRINT [2, 0] "NEW YORK"
250 PRINT
260 PRINT [2, 0] "RIO DE JANEIRO"
270 PRINT
280 PRINT [2, 0] "DAKAR"
290 PRINT
300 PRINT [7, 0] "** HAMBURG **"
310 PRINT
320 PRINT [4, 2] "KAPSTADT"
330 PRINT
340 PRINT [4, 2] "MOSKAU"
350 PRINT
360 PRINT [4, 2] "HONGKONG"
370 PRINT
380 PRINT [4, 2] "TOKIO"
390 PRINT
400 PRINT [4, 2] "SYDNEY"
.
.
```

Sollten Ihre Ergänzungen nicht richtig sein, kehren Sie zurück zur → **LE 16.3** !

LE 16.4

Zur Gestaltung von grafischen Darstellungen auf dem Bildschirm stehen die Befehle

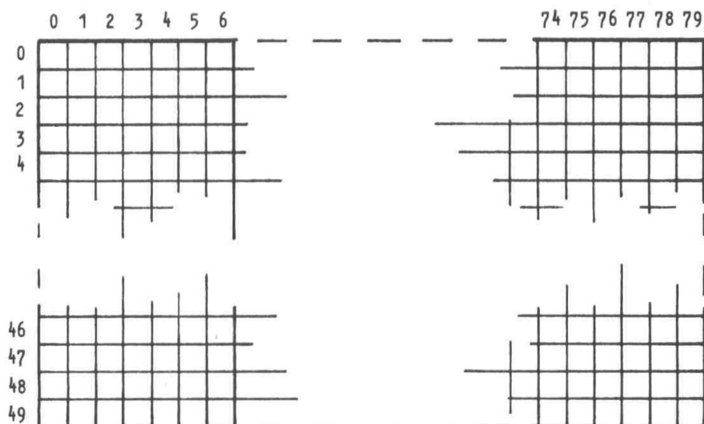
SET (von engl. "to set" = setzen)

und

RESET (von engl. "to reset" = zurücksetzen)

zur Verfügung, und zwar kann man mit SET an einer zu spezifizierenden Stelle einen Punkt "setzen" und mit RESET ggf. wieder löschen.

Abweichend von der sonst üblichen Einteilung des Bildschirms (40 Spalten / 25 Zeilen) besteht für SET/RESET ein feineres Raster von 80 Spalten (Spalte 0 - Spalte 79) und 50 Zeilen (Zeile 0 - Zeile 49):



Jede der auf der vorangegangenen Seite skizzierten Stellen läßt sich mit zwei Angaben genau lokalisieren, nämlich mit der

- Nennung der Spalte und der

- Nennung der Zeile.

Wenn wir nun nach dem Befehlswort SET einen Wert zwischen 0 und 79 (für die Spalte) und nach einem dann zu setzenden Komma einen Wert zwischen 0 und 49 (für die Zeile) - sei es als Variable, sei es als Konstante - angeben, so bestimmen wir damit eine Stelle, an der ein Punkt gesetzt werden soll.

Darüber hinaus ist es möglich, nach einem Komma noch einen Farbwert (0 - 7) für den Punkt zu bestimmen. Wenn man den Farbwert nicht angibt, verwendet das System die gerade aktuelle Zeichenfarbe. (Die Liste der Farbwerte haben wir bereits auf Seite 16-03 aufgeführt.)

Für SET bestehen demnach zwei Formate, nämlich

(I) SET S, Z

(Spalte: 0 - 79) (Zeile: 0 - 49)

(II) SET S, Z, F

(Farbwert: 0 - 7)

Das Gegenstück zu SET S, Z bildet die Anweisung

RESET S, Z ,

mit der an der aufgeführten Stelle ein gesetzter Punkt gelöscht wird.

Das folgende Programm soll lediglich dazu dienen, sich mit der Funktionsweise von S E T / R E S E T vertraut zu machen. Da ein Programmende nicht vorgesehen ist, muß die Ausführung ggf. durch gleichzeitige Betätigung von [SHIFT] und [BREAK] unterbrochen werden.

Programmliste:

```
10 REM *********************
20 REM * SET/RESET-BEISPIEL *
30 REM *********************
40 CLS
50 REM BESCHRIFTUNG
60 REM -----------
70 COLOR ,, 2, 0
80 PRINT TAB(5) " S   P   A   L   T   E
  N "
90 CURSOR 0, 5
100 PRINT " "
110 PRINT "Z"
120 PRINT " "
130 PRINT " "
140 PRINT "E"
150 PRINT " "
160 PRINT " "
170 PRINT "I"
180 PRINT " "
190 PRINT " "
200 PRINT "L"
210 PRINT " "
220 PRINT " "
230 PRINT "E"
240 PRINT " "
250 PRINT " "
260 PRINT "N"
270 PRINT " "
280 REM ZEICHNEN DES SPALTEN-STRICHS
290 REM ---------------------------
300 FOR X = 10 TO 79
310 SET X, 10, 0
320 NEXT X
```

```
330 REM ZEICHEN DER SPALTEN-EINTEILUNG
340 REM -----------------------------
350 FOR X = 10 TO 70 STEP 10
360 SET X, 9, 0
370 NEXT X
380 CURSOR 4, 3
390 PRINT "10"
400 CURSOR 9, 3
410 PRINT "20"
420 CURSOR 14, 3
430 PRINT "30"
440 CURSOR 19, 3
450 PRINT "40"
460 CURSOR 24, 3
470 PRINT "50"
480 CURSOR 29, 3
490 PRINT "60"
500 CURSOR 34, 3
510 PRINT "70"
520 REM ZEICHNEN DES ZEILEN-STRICHS
530 REM --------------------------
540 FOR Y = 10 TO 49
550 SET 10, Y, 0
560 NEXT Y
570 REM ZEICHNEN DER ZEILEN-EINTEILUNG
580 REM -----------------------------
590 FOR Y = 10 TO 40 STEP 10
600 SET 9, Y, 0
610 NEXT Y
620 CURSOR 2, 5
630 PRINT "10"
640 CURSOR 2, 10
650 PRINT "20"
660 CURSOR 2, 15
670 PRINT "30"
680 CURSOR 2, 20
690 PRINT "40"
700 COLOR ,, 7, 1
```

```
710 REM SETZEN DER PUNKTE GEM. EINGABE
720 REM DES ANWENDERS
730 REM -----------------------------
740 CURSOR 7, 23
750 PRINT "                         "
760 CURSOR 7, 23
770 INPUT "SPALTE ? "; X
780 CURSOR 7, 23
790 PRINT "                         "
800 CURSOR 7, 23
810 INPUT "ZEILE ? "; Y
820 CURSOR 7, 23
830 PRINT "                         "
840 CURSOR 7, 23
850 INPUT "FARBE (0-7) ? "; F
860 CURSOR 7, 23
870 PRINT "                         "
880 SET X, Y, F
890 CURSOR 7, 23
900 PRINT "O.K. ? (J/N)";
910 GET JN$
920 IF JN$ = "J" THEN 710
930 IF JN$ = "N" THEN 950
940 GOTO 910
950 RESET X, Y
960 GOTO 710
```

PE 16.4

(1) In Spalte 50 / Zeile 30 des Bildschirms ist ein Punkt mit der gerade aktuellen Zeichenfarbe zu setzen. Wie lautet die Anweisung?

SET 50,30 ....

(2) In Spalte 60 / Zeile 45 des Bildschirms ist ein schwarzer Punkt (Farbwert: 0) zu setzen. Wie lautet die Anweisung?

SET 60,45,0 ....

(3) Der in Spalte 60 / Zeile 45 gesetzte Punkt ist wieder zu löschen. Wie lautet die Anweisung?

RESET 60,45 ....

(4) Unter Verwendung einer FOR...NEXT-Schleife (vgl. hierzu Lektion 20) ist mit Hilfe der SET-Anweisung in der Bildschirmzeile 10 von Spalte 5 bis Spalte 70 ein waagerechter, schwarzer (gepunkteter) Strich zu ziehen. Ergänzen Sie hierzu die fehlenden Angaben in der Befehlszeile 110!

```
.
.
.
100 FOR S = 5 TO 70

110 SET S, ...............

120 NEXT S
.
.
.
```

(5) Unter Verwendung einer FOR...NEXT-Schleife (vgl. hierzu Lektion 20) ist mit Hilfe der SET-Anweisung in der Bildschirmspalte 20 von Zeile 10 bis Zeile 30 ein senkrechter, schwarzer (gepunkteter) Strich zu ziehen. Ergänzen Sie hierzu die fehlenden Angaben in der Befehlszeile 210!

```
.
.
.
200 FOR Z = 10 TO 30

210 SET .................

220 NEXT Z
.
.
.
```

LÖS 16.4

(1) In Spalte 50 / Zeile 30 des Bildschirms ist ein Punkt mit der gerade aktuellen Zeichenfarbe zu setzen. Die Anweisung lautet:

.. SET 50, 30 .

(2) In Spalte 60 / Zeile 45 des Bildschirms ist ein schwarzer Punkt (Farbwert: 0) zu setzen. Die Anweisung lautet:

.. SET 60, 45, 0 .

(3) Der in Spalte 60 / Zeile 45 gesetzte Punkt ist wieder zu löschen. Die Anweisung lautet:

.. RESET 60, 45 .

(4) Unter Verwendung einer FOR... NEXT-Schleife ist mit Hilfe der SET-Anweisung in der Bildschirmzeile 10 von Spalte 5 bis Spalte 70 ein waagerechter, schwarzer (gepunkteter) Strich zu ziehen. Der ergänzte Programmteil hat folgendes Aussehen:

```
100 FOR S = 5 TO 70
110 SET S, 10, 0
120 NEXT S
```

(5) Unter Verwendung einer FOR... NEXT-Schleife ist mit Hilfe der SET-Anweisung in der Bildschirmspalte 20 von Zeile 10 bis Zeile 30 ein senkrechter, schwarzer (gepunkteter) Strich zu ziehen. Der ergänzte Programmteil hat folgendes Aussehen:

```
200 FOR Z = 10 TO 30
210 SET 20, Z, 0
220 NEXT Z
```

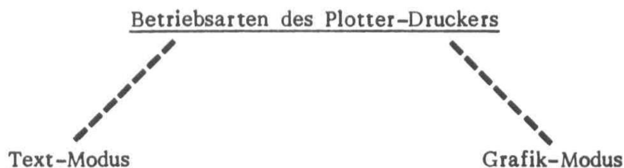
Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 16.4** !

Plotter

LE 16.5

Nachdem wir die wichtigsten Anweisungen kennengelernt haben, mit denen man die für den Bildschirm bestimmten Ausgaben gestalten kann, wollen wir nun die spezifischen Befehle für den Plotter-Drucker skizzieren und deren Anwendung in den folgenden **ÜBUNG**sprogrammen vertiefen.

Für den Plotter-Drucker gibt es zwei Betriebsarten:



Dementsprechend gibt es Befehle für den Plotter-Drucker, die

(1) im Text-Modus u n d im Grafik-Modus,

(2) n u r im Text-Modus,

(3) n u r im Grafik-Modus

verwendet werden dürfen.

Wenn der Programmierer nichts anderes bestimmt hat, befindet sich das System immer im Text-Modus.

Mit Hilfe der Anweisung

MODE GR (von engl. "mode: graphics = Erscheinungsform/Betriebsart: Grafik)

schaltet man den Plotter-Drucker in den Grafik-Modus (um anschließend die für diese Betriebsart vorgesehenen Befehle zu nutzen) und die Verwendung von

MODE TN oder

MODE TL oder

MODE TS

(vgl. hierzu die Ausführungen auf Seite 12-11)

bewirkt die Rückkehr zum Text-Modus.

Hinweis:

Wenn der Plotter-Drucker für einen längeren Zeitraum nicht benutzt werden soll, so empfehlen wir, die Minen in Anlehnung an die Gebrauchsanweisung des Herstellers aus dem Gerät zu nehmen, sie mit Kappen zu versehen und im verschlossenen Behälter aufzubewahren. Man verhindert dadurch ein vorzeitiges Austrocknen.

PE 16.5

(Bitte, die linke Seite abdecken!)

Welche der folgenden Behauptungen ist/sind richtig?

A Für den Plotter-Drucker gibt es zwei Betriebsarten, nämlich den Text-Modus und den Grafik-Modus.

B Will man für den Plotter-Drucker jene für den Grafik-Modus vorgesehenen Befehle verwenden, muß das Gerät zuvor in einer Fachwerkstatt umgerüstet werden.

C Mit Hilfe der Anweisung MODE GR kann man den Plotter-Drucker in den Grafik-Modus umschalten, wodurch spezielle, für diese Betriebsart vorgesehene Befehle genutzt werden können.

D Für den Plotter-Drucker ist vom Hersteller absichtlich kein Grafik-Modus vorgesehen, weil Zeichnungen sinnvollerweise nur von großen Künstlern erstellt werden sollten.

E Die Verwendung von MODE TN oder MODE TL oder MODE TS bewirkt die Rückkehr vom Grafik-Modus zum Text-Modus.

Der/die Lösungsbuchstabe/n lautet/lauten

....

LÖS 16.5

Die Lösungsbuchstaben lauten

A, C, E .

Sollten Ihre Lösungsbuchstaben nicht richtig sein, kehren Sie zurück zur
→ **LE 16.5** !

LE 16.6

In der vorangegangenen **LE** haben wir gelernt, daß man den Plotter-Drucker mit M O D E G R in den Grafik-Modus umschaltet und mit M O D E T N oder M O D E T L oder M O D E T S eine Rückkehr zum Text-Modus herbeiführt.

Folgende Instruktionen stehen im Hinblick auf die Betriebsarten zur Verfügung:

(1) Befehl für Text-Modus u n d Grafik-Modus

PCOLOR N

Diese Anweisung bestimmt die Druckfarbe des Plotter-Druckers:

N = Wert von 0 - 3, und zwar 0 = schwarz, 1 = blau, 2 = grün, 3 = rot.

(vgl. hierzu auch die Ausführungen auf Seite 12-11)

(2) Befehle n u r für Text-Modus

Wenn einer der folgenden, ausschließlich für einen bestimmten Modus vorgesehenen Befehle in der falschen Betriebsart verwendet wird - d. h. eine für den Text-Modus reservierte Anweisung im Grafik-Modus oder eine für den Grafik-Modus reservierte Anweisung im Text-Modus - unterbricht das System die Programmausführung mit der Fehlermeldung "Printer mode error in .. (= Zeile)".

(a) TEST (von engl. "to test" = überprüfen)

Dieser Befehl dient der Überprüfung der Funktionstüchtigkeit der vier Farbminen des Plotter-Druckers durch Ausgabe von

vier Quadraten:

0	1	2	3	(Wert N bei PCOLOR N)
schwarz	blau	grün	rot	

Wenn die Minen einen längeren Zeitraum nicht genutzt wurden, kann man durch mehrmalige Wiederholung des Befehls u. U. eine "Aktivierung des Farbflusses" bewirken. Im Regelfall wird T E S T nur in der direkten Betriebsart (d. h. ohne Zeilennummer) eingesetzt (vgl. hierzu die Ausführungen in der Fußnote auf Seite 6-07).

(b) <u>SKIP N</u> (von engl. "to <u>skip</u>" = springen, überschlagen)

Mit S K I P kann das Papier des Plotter-Druckers bis maximal 20 Zeilen vorwärts oder rückwärts transportiert werden. Die Anzahl der Zeilen ergibt sich aus dem nach dem Befehlswort S K I P anzugebenden Wert (Konstante oder Variable), der zwischen -20 und +20 liegen muß.

<u>Testprogramm:</u>

```
10 REM *************
20 REM * SKIP-TEST *
30 REM *************
40 CLS
50 PRINT "WIEVIEL ZEILEN VOR ODER ZURUEC
K?"
60 PRINT "(GRENZEN: -20 BIS +20)"
70 INPUT X
80 IF (X < -20) + (X > +20) THEN PRINT "      *
EINGABEFEHLER!" : GOTO 50
90 SKIP X
```

* vgl. hierzu die Fußnote auf Seite 16-20

(c) PAGE N (von engl. "page" = Seite)

Diese Anweisung legt die Zahl der Zeilen fest, die ohne Zwischenraum auszugeben sind, um dann automatisch eine Leerzeile hinzuzufügen. Anders ausgedrückt: Die auf dem Endlospapier gedruckten Ergebnisse werden automatisch durch eine Leerzeile getrennt, wenn N Zeilen geschrieben worden sind, weil dadurch für Ablagezwecke ein einfacheres Zerschneiden der Papierbahn möglich ist.

Der nach dem Befehlswort PAGE anzugebende Wert (Konstante oder Variable) darf zwischen 1 und 72 liegen.

Testprogramm:

```
10 REM *************
20 REM * PAGE-TEST *
30 REM *************
40 CLS
50 PRINT "ZEILEN PRO SCHREIBBLOCK ?"
60 PRINT "(1 - 72)"
70 INPUT X
80 IF (X < 1) + (X > 72) THEN PRINT "EIN
GABEFEHLER!" : GOTO 50
90 PAGE X
100 FOR I = 1 TO 30
110 PRINT/P "TESTZEILE"; I
120 NEXT I
130 REM WIEDERHERSTELLUNG DER UEBLICHEN
140 REM PAGE-EINSTELLUNG
150 REM ----------------------------
160 PAGE 72
```

Ergebnisse eines Programmlaufs:

(Ausgaben auf dem Plotter-Drucker, wenn aufgrund des Befehls in Zeile 70 der Wert 6 eingegeben wird.)

```
TESTZEILE 1
TESTZEILE 2
TESTZEILE 3
TESTZEILE 4
TESTZEILE 5
TESTZEILE 6

TESTZEILE 7
TESTZEILE 8
TESTZEILE 9
TESTZEILE 10
TESTZEILE 11
TESTZEILE 12

TESTZEILE 13
TESTZEILE 14
TESTZEILE 15
TESTZEILE 16
TESTZEILE 17
TESTZEILE 18

TESTZEILE 19
TESTZEILE 20
TESTZEILE 21
TESTZEILE 22
TESTZEILE 23
TESTZEILE 24

TESTZEILE 25
TESTZEILE 26
TESTZEILE 27
TESTZEILE 28
TESTZEILE 29
TESTZEILE 30
```

(d) LIST/P

(vgl. hierzu die Ausführungen auf Seite 8-24 und 6-05)

Hinweis: Grafische Zeichen und Symbole werden als ASCII-Werte in hexadezimaler Form und in blauer Farbe gedruckt.

(e) PRINT/P

(vgl. hierzu die Ausführungen auf Seite 12-11)

Hinweis: Grafische Zeichen und Symbole werden als ASCII-Werte in hexadezimaler Form und in blauer Farbe gedruckt.

(f) PRINT/P USING

Die Anweisung P R I N T / P U S I N G entspricht dem erst in Lektion 26 darzustellenden Befehl P R I N T U S I N G - die Ausgabe erfolgt jedoch auf dem Plotter-Drucker.

Die bisher erläuterten Befehle, die

(1) im Text-Modus u n d Grafik-Modus und

(2) n u r im Text-Modus

erlaubt sind, bilden die Grundlage für die Aufgaben der folgenden **PE 16.6**. Auf die Instruktionen, die

(3) n u r im Grafik-Modus

eingesetzt werden dürfen, wollen wir dann in **LE 16.7** eingehen.

PE 16.6

(1) Welche der folgenden Behauptungen ist/sind richtig?

A Nachdem ein Unbefugter mit einem SHARP MZ-700 gespielt hat, erscheinen die Ausdrucke des Plotter-Druckers nur noch in Rot; zur Umstellung auf Schwarz müßte eine Fachwerkstatt aufgesucht werden.

B Die Ausgaben auf dem Plotter-Drucker können in vier verschiedenen Farben erfolgen. Ein Wechsel der Farbe wird mit der Anweisung COLOR ,, ZF, HF herbeigeführt.

C Mit dem Befehl PCOLOR N , der im Text-Modus und im Grafik-Modus verwendet werden darf, bestimmt man die Druckfarbe des Plotter-Druckers, wobei N einen numerischen Wert von 0 (= schwarz), 1 (= blau), 2 (= grün) oder 3 (= rot) darstellt.

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

(2) Wie lautet der Befehl zur Überprüfung der Funktionstüchtigkeit der vier Farbminen des Plotter-Druckers?

. . . .

(3) Wie lautet der Befehl, aufgrund dessen die auf dem Endlospapier gedruckten Ergebnisse nach jeweils 50 Zeilen automatisch durch eine Leerzeile voneinander abgehoben werden?

. . . .

(4) (Ergänzen Sie die folgenden Ausführungen an den gekennzeichneten Stellen!)

Mit SKIP kann das Papier des Plotter-Druckers bis maximal

. . . . Zeilen vorwärts oder rückwärts transportiert werden. Die Anzahl der Zeilen ergibt sich aus dem nach dem Befehlswort SKIP anzugebenden Wert (Konstante oder

Variable), der zwischen . . . . und . . . . liegen muß.

(5) Sie wollen den SHARP MZ-700 wie einen normalen Taschenrechner benutzen und in der direkten Betriebsart das Ergebnis für 135,50 + (14 x 1,355) errechnen und auf dem Plotter-Drucker ausgeben lassen. Wie lautet der Befehl (ohne Zeilennummer!)?

. . . .

(6) Das im Arbeitsspeicher befindliche Programm soll mit roter Druckfarbe auf dem Plotter-Drucker ausgegeben werden. Wie lauten die beiden Kommandos, die man zu diesem Zweck nacheinander erteilen muß?

. . . .

. . . .

LÖS 16.6

(1) Der Lösungsbuchstabe lautet

C .

(2) Der Befehl zur Überprüfung der Funktionstüchtigkeit der vier Farbminen des Plotter-Druckers lautet

T E S T .

(3) Der Befehl, aufgrund dessen die auf dem Endlospapier gedruckten Ergebnisse nach jeweils 50 Zeilen automatisch durch eine Leerzeile voneinander abgehoben werden, lautet

P A G E 5 0 .

(4) Mit S K I P kann das Papier des Plotter-Druckers bis maximal 20 Zeilen vorwärts oder rückwärts transportiert werden. Die Anzahl der Zeilen ergibt sich aus dem nach dem Befehlswort S K I P anzugebenden Wert (Konstante oder Variable), der zwischen - 2 0 und + 2 0 liegen muß.

(5) Sie wollen den SHARP MZ-700 wie einen normalen Taschenrechner benutzen und in der direkten Betriebsart das Ergebnis für 135,50 + (14 x 1,355) errechnen und auf dem Plotter-Drucker ausgeben lassen. Der entsprechende Befehl lautet

P R I N T / P 1 3 3 . 5 0 + (1 4 * 1 . 3 5 5) .

(Beachten Sie die Notation in BASIC: ein Punkt (statt eines Kommas) zur Kennzeichnung der Dezimalstelle und ein Stern als Multiplikationszeichen!)

(6) Das im Arbeitsspeicher befindliche Programm soll mit roter Druckfarbe auf dem Plotter-Drucker ausgegeben werden. Die Kommandos lauten

P C O L O R 3 und

L I S T / P .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 16.6** !

LE 16.7

Bereits in **LE 16.5** hatten wir gelernt, daß mit M O D E G R für den Plotter-Drucker der Grafik-Modus eingeschaltet wird. Nach Erteilung des Befehls können spezielle, besonders mächtige Grafik-Anweisungen genutzt werden. Danach ist es im Regelfall empfehlenswert, in den Text-Modus zurückzukehren.

B e i s p i e l :

```
.
.
.
240 MODE GR
250 ...
260 ...
270 ...
280 ...
290 ...
300 ...
310 ...
320 ...
330 ...
340 ...
350 MODE TN
.
.
.
```

Erläuterungen:

240 → Einschalten des Grafik-Modus

250 bis 340:

(a) vor allem Grafik-Anweisungen

(b) ggf. auch alle übrigen Befehle

(c) n i c h t zulässig: Befehle für den Plotter-Drucker, für die der Text-Modus erforderlich ist

350 → Rückkehr zum Text-Modus (auch möglich mit: M O D E T L oder M O D E T S)

In der vorangegangen **LE 16.6** haben wir

(1) den Befehl für Text-Modus u n d Grafik-Modus, nämlich

P C O L O R N	(zum Bestimmen der Druckfarbe, wobei N = 0, 1, 2 oder 3),

(2) die Befehle n u r für Text-Modus, und zwar

T E S T	(zur Überprüfung der Funktionstüchtigkeit der Farbminen),
S K I P N	(zum Vorwärts- und Rückwärtstransport des Papiers, wobei N zwischen -20 und +20 liegen muß),
P A G E N	(zur Festlegung der Zeilen pro Schreibblock, wobei für N ein Wert zwischen 1 und 72 anzugeben ist),
L I S T / P	(zum Auflisten des Programms oder von Programmteilen),
P R I N T / P	(für Ausgaben) und
P R I N T / P U S I N G	(für formatierte Ausgaben von numerischen Werten; vgl. hierzu Lektion 26)

erläutert und beschreiben nun

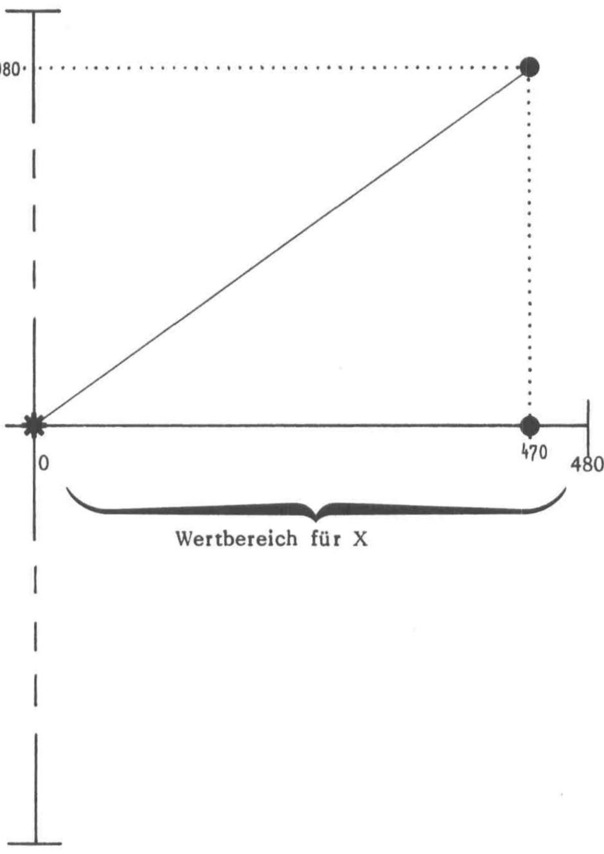
(3) Befehle n u r für Grafik-Modus .

(a) LINE (von engl. "line" = Linie, Strich)

Diese Anweisung dient zum Zeichnen von geraden Linien. Hierbei stehen mehrere Formate zur Verfügung:

Format 1: L I N E X , Y

Mit Hilfe der Werte von X und Y bestimmt man, zu welcher Stelle des Papiers - ausgehend von der aktuellen Position des Schreibkopfs - ein Strich gezogen werden soll. Aus der folgenden Darstellung ergeben sich die zulässigen Werte:



Die zulässigen Werte für X liegen demnach im Bereich von 0 - 480 und jene für Y im Bereich von +999 - -999 .

Das durch den Stern und die zwei Punkte gekennzeichnete Dreieck läßt sich also beispielsweise durch folgende Positionsangaben (X, Y) definieren: 0, 0 - 470, 980 - 470, 0 .

Zu Beginn des Wechsels zum Grafik-Modus befindet sich der Schreibkopf in der Position 0, 0 . Angenommen, man wolle einen Strich vom Ursprung (0, 0) zur Position 470, 980 ziehen, also von der durch einen Stern gekennzeichneten Stelle zum rechten oberen Punkt, so lautet die Anweisung

LINE 470, 980 1) .

Mit LINE 470, 0

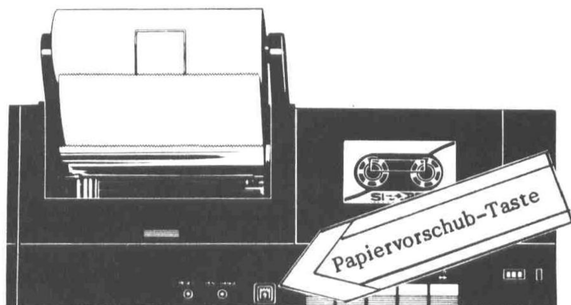
ziehen wir dann eine senkrechte Linie zur (gedachten) X-Achse und

mit LINE 0, 0

einen Strich (auf der gedachten X-Achse) zur Ausgangsposition.

---

1) a) Es muß darauf geachtet werden, daß mit Hilfe der Papiervorschub-Taste mindestens 25 cm Papier aus der Austrittsöffnung herausragt, damit nicht versehentlich die Walze beschrieben wird.



b) Vor Erteilung dieses Befehls muß mit MODE GR ein Wechsel zum Grafik-Modus erfolgt sein.

c) Die Befehle können per Programm oder - wie bei fast allen BASIC-Anweisungen - auch in der direkten Betriebsart (also ohne Zeilennummer) erteilt werden.

Die drei LINE-Anweisungen dürfen auch zu einer einzigen Instruktion zusammengefaßt werden, indem nach dem Befehlswort LINE alle jene Positionen aufgeführt werden, die - ausgehend von der Ausgangsstellung - nacheinander durch Striche zu verbinden sind:

Format 2: LINE X1,Y1, X2,Y2, X3,Y3 . . .

Darüber hinaus ist es möglich, nach dem Befehlswort LINE zunächst einen Linientyp zu spezifizieren, und zwar mit einem Wert von 1 - 16 (als Variable oder Konstante), dem ein Prozentzeichen (%) vorangestellt sein muß:

Format 3: LINE %LT, X1,Y1, X2,Y2, X3,Y3 . . .

(Wert von 1 - 16)

Der folgende Programmausdruck zeigt die Möglichkeiten der Liniengestaltung:

```
BEIM PLOTTER-DRUCKER DES
SHARP MZ-700 STEHEN DIE
FOLGENDEN LINIENTYPEN
ZUR VERFUEGUNG:
========================

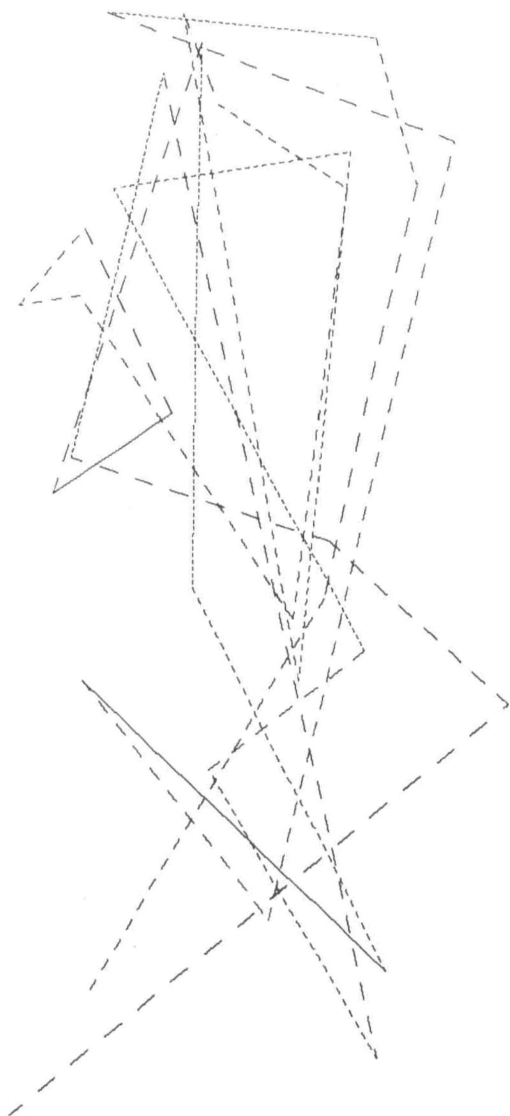
%LT = 1 :  ____________________________
%LT = 2 :  ............................
%LT = 3 :  ----------------------------
%LT = 4 :  ----------------------------
%LT = 5 :  - - - - - - - - - - - - - -
%LT = 6 :  - - - - - - - - - - - - - -
%LT = 7 :  - - - - - - - - - - - - - -
%LT = 8 :  - - - - - - - - - - - - - -
%LT = 9 :  - - - - - - - - - - - - - -
%LT = 10 : _ _ _ _ _ _ _ _ _ _ _ _ _ _
%LT = 11 : _ _ _ _ _ _ _ _ _ _ _ _ _ _
%LT = 12 : _ _ _ _ _ _ _ _ _ _ _ _ _
%LT = 13 : _ _ _ _ _ _ _ _ _ _ _ _ _
%LT = 14 : _ _ _ _ _ _ _ _ _ _ _ _
%LT = 15 : _ _ _ _ _ _ _ _ _ _ _ _
%LT = 16 : _ _ _ _ _ _ _ _ _ _ _
```

Programmliste (zum Ausdruck auf Seite 16-50):

```
10 REM ******************************
20 REM *   AUSDRUCK DER LINIENTYPEN   *
30 REM ******************************
40 PRINT/P "BEIM PLOTTER-DRUCKER DES"
50 PRINT/P "SHARP MZ-700 STEHEN DIE"
60 PRINT/P "FOLGENDEN LINIENTYPEN"
70 PRINT/P "ZUR VERFUEGUNG:"
80 PRINT/P "======================"
90 PRINT/P
100 FOR LT = 1 TO 16
110 MODE TN
120 PRINT/P
130 PRINT/P "%LT ="; LT; " :";
140 MODE GR
150 MOVE 150, 0
160 LINE %LT, 480, 0
170 NEXT LT
180 MODE TN
190 END
```

Nachdem wir die Formate 1 - 3 zur LINE-Anweisung kennengelernt haben, wollen wir zunächst ein Anwendungsbeispiel wiedergeben. Mit Hilfe des auf Seite 16-53 ff. abgedruckten Programms kann man ein Kunstwerk (bzw. was in der heutigen Zeit als solches bezeichnet werden kann) aus vierfarbigen Linien verschiedenen Typs zeichnen lassen. Die Kopie eines besonders kostbaren Exemplars (aus drucktechnischen Gründen nur in Schwarz-Weiß) ist auf der folgenden Seite zu finden. Erläuterungen zum Programm erübrigen sich, da es in ausführlicher Weise mit REM-Befehlen kommentiert ist.

Kunstwerk aufgrund des Programms auf Seite 16-53 ff.:



Programmliste:

```
10 REM ******************************
20 REM * ERSTELLUNG EINES KUNSTWERKS  *
30 REM * AUS MEHREREN, UNTERSCHIED-   *
40 REM * LICH GESTRICHTELTEN LINIEN   *
50 REM * IN VIER VERSCHIEDENEN FARBEN *
60 REM ******************************
70 CLS
80 REM AUFFORDERUNG ZUM PAPIERVORSCHUB,
90 REM DANN WARTESCHLEIFE, BIS
100 REM ERLEDIGUNG BESTAETIGT IST
110 REM ----------------------------
120 PRINT [2, 0] "ACHTUNG: PAPIER MIND."
130 PRINT
140 PRINT [2, 0] "25 CM VORSCHIEBEN!"
150 PRINT : PRINT : PRINT : PRINT
160 PRINT TAB(20) "ERLEDIGT ? (J/N)"
170 GET A$
180 IF A$ = "J" THEN 200
190 GOTO 170
200 CLS
210 REM EINGABE DER ANZAHL DER ZU
220 REM ZEICHNENDEN STRICHE
230 REM ----------------------
240 PRINT "ANZAHL DER GEWUENSCHTEN STRIC
HE ?"
250 PRINT "(MAXIMAL 50 ! )"
260 PRINT
270 INPUT E
280 IF (E < 0) + (E > 50) THEN PRINT "EI
NGABEFEHLER!" : GOTO 240
290 REM ============================
300 REM ERSTELLUNG DES KUNSTWERKS
310 REM ***********************
320 MODE GR
330 REM → → → SCHLEIFENBEGINN ← ← ←
340 REM ------------------------
350 FOR I = 1 TO E
360 REM ERMITTLUNG EINER ZAHL VON
370 REM 0 - 3 (FARBE) MIT HILFE
380 REM DER ZUFALLSFUNKTION RND(X)
390 REM ----------------------
400 LET F = INT(4 * RND(1))
410 REM
```

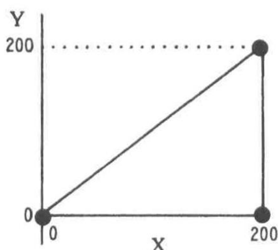
```
420 PCOLOR F
430 REM ERMITTLUNG EINER ZAHL VON
440 REM 0 - 480 (X-WERT) MIT HILFE
450 REM DER ZUFALLSFUNKTION RND(X)
460 REM --------------------------
470 LET X = INT(481 * RND(1))
480 REM
490 REM ERMITTLUNG EINER ZAHL VON
500 REM 0 - 999 (Y-WERT) MIT HILFE
510 REM DER ZUFALLSFUNKTION RDN(X)
520 REM --------------------------
530 LET Y = INT(1000 * RND(1))
540 REM
550 REM ERMITTLUNG EINER ZAHL VON
560 REM VON 1 - 16 (LINIENTYP) M. HILFE
570 REM DER ZUFALLSFUNKTION RND(X)
580 REM -------------------------------
590 LET LT  = INT(16 * RND(1)+1)
600 LINE %LT, X, Y
610 NEXT I
620 REM → → → SCHLEIFENENDE ← ← ←
630 REM -------------------------
640 REM ===============================
650 REM RUECKKEHR ZUM TEXT-MODUS
660 REM U. SCHWARZE FARBWAHL
670 REM ------------------------
680 MODE TN
690 PCOLOR 0
700 REM ENDE-NACHRICHT
710 REM ------------
720 CLS
730 PRINT : PRINT : PRINT : PRINT
740 PRINT "DAS KUNSTWERK WURDE SOEBEN"
750 PRINT
760 PRINT "FERTIGGESTELLT!"
770 END
```

Große Ähnlichkeit mit der LINE-Anweisung besitzt

(b) RLINE (von engl. "relative line" = relativer Strich).

Während aber bei LINE die Positionsangaben immer auf den (Koordinaten-)Ursprung bezogen sind, ist für RLINE die aktuelle Druckkopfposition Ausgangspunkt für die Berechnung der nächsten anzusteuernden Stelle.

Beispiel:



Das durch die drei Punkte gekennzeichnete Dreieck könnte mit folgenden LINE-Anweisungen gezeichnet werden:

```
LINE 200,0

LINE 200,200

LINE 0,0
```

Mit RLINE müßte man hingegen folgendermaßen vorgehen:

RLINE 200,0 (kein Unterschied zu LINE)

RLINE 0,200 (Ausgangspunkt ist jetzt der mit der ersten RLINE-Anweisung angesteuerte Punkt. Von ihm aus betrachtet bleibt der X-Wert gleich und nur der Y-Wert ist um 200 zu verändern.)

RLINE -200,-200 (Ausgangspunkt ist nun der mit dem zweiten RLINE-Befehl angesteuerte Punkt. Von ihm aus gesehen verändert sich der X-Wert um -200 und auch der Y-Wert ist mit -200 anzusetzen.)

Die für RLINE zur Verfügung stehenden Formate entsprechen jenen von LINE; allerdings war es natürlich erforderlich, die zulässigen X-Werte auf den negativen Bereich auszudehnen:

Format 1: RLINE X,Y
(zulässige Werte für
X: -480 bis 480,
Y: -999 bis 999)

Format 2: RLINE X1,Y1, X2,Y2, X3,Y3 ...
(zulässige Werte für
X1, X2, X3, ... Xn: -480 bis 480,
Y1, Y2, Y3, ... Yn: -999 bis 999)

Format 3: RLINE %LT, X1,Y1, X2,Y2, X3,Y3...
(zulässige Werte für
LT: 1 bis 16,
X1, X2, X3, ... Xn: -480 bis 480,
Y1, Y2, Y3, ... Yn: -999 bis 999)

ACHTUNG:

Für die obigen und die folgenden Anweisungen ist unbedingt zu beachten, daß der zulässige Druckbereich nicht überschritten wird; andernfalls können Störungen verursacht werden!

Will man von einem Punkt zu einem anderen wechseln, ohne daß ein Strich gezogen werden soll, so verwendet man

(c) MOVE X, Y (von engl. "to move" = bewegen),

wobei die X-Werte zwischen -480 und 480 und die Y-Werte zwischen -999 und 999 liegen dürfen. Die Positionsangaben beziehen sich auf den (Koordinaten-) Ursprung, während bei

(d) RMOVE X, Y (von engl. "relative move" = relative Bewegung)

die aktuelle Druckkopfposition den Ausgangspunkt für die Berechnung der nächsten anzusteuernden Stelle bildet. Im übrigen gelten für R M O V E die gleichen Wertgrenzen wie für M O V E.

(e) PHOME (von engl. "pen to home position" = Mine zurück zur Ausgangsposition)

führt den Druckkopf in die Ausgangsposition zurück.

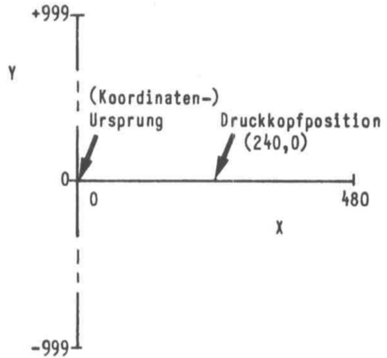
Es wird aufgefallen sein, daß sowohl für M O V E als auch für R M O V E negative X-Werte zulässig sind. Dieser Umstand ist nur verständlich im Zusammenhang mit

(f) HSET (von engl. "to hold and to set" = halten und festsetzen).

Diese Anweisung bietet die Möglichkeit, eine beliebige (aktuelle) Druckkopfposition als (Koordinaten-) Ursprung zu definieren.

Beispiel:

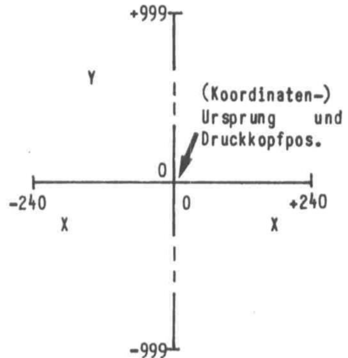
	Erläuterungen:
.	
.	
.	
100 MODE GR	Wechsel zum Grafik-Modus
110 MOVE 240, 0	Der Druckkopf wird zur Mitte der X-Achse bewegt. Noch gelten folgende Werte für das Koordinatensystem:



120 HSET

Hiermit wird die aktuelle Druckkopfposition zum neuen (Koordinaten-)Ursprung bestimmt, so daß nun folgende Werte gelten:

.
.
.
.
.
.
.
.
.
.
.
.
.



250 MODE TN

Mit der Rückkehr in den Text-Modus werden die durch HSET erfolgten Festlegungen gelöscht, so daß nach

.
.
.

350 MODE GR

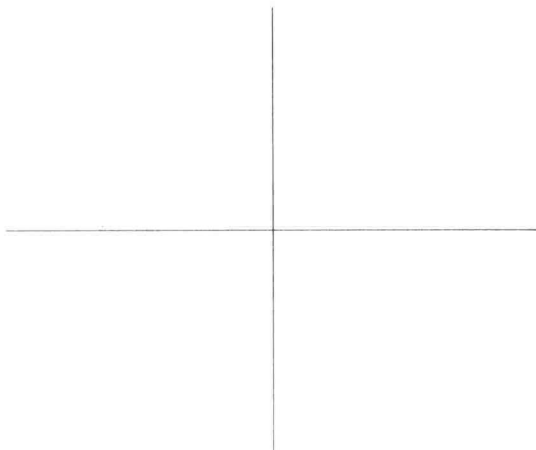
abermaligem Wechsel zum Grafik-Modus das ursprüngliche Koordinatensystem wieder gültig ist.

.
.
.

Mit Hilfe von L I N E und M O V E kann man selbstverständlich u. a. auch ein Koordinatenkreuz zeichnen, z. B. folgendermaßen:

```
.
.
.
100 MODE TN              Wechsel zum Text-Modus
110 SKIP 20              Papiervorschub
120 MODE GR              Wechsel zum Grafik-Modus
130 LINE 480, 0          Zeichnen einer waagerechten Linie (X-Achse)
                         von links nach rechts
140 MOVE 240, 200        Transport des Schreibkopfs in die Mitte der
                         Papierbahn ca. 4 cm unterhalb des waage-
                         rechten Strichs
150 LINE 240, 200        Zeichnen einer senkrechten Linie (Y-Achse)
160 PHOME                Transport des Schreibkopfs in die Ausgangs-
                         position
.
.
.
```

Das aufgrund der obigen Befehle erstellte Koordinatenkreuz hat folgendes Aussehen:



Zum Zeichnen von Koordinatenkreuzen steht beim SHARP MZ-700 zusätzlich eine Spezialanweisung zur Verfügung:

(g) AXIS XY, S, Z (von engl. "axis" = Achse(nkreuz)).

- Z: bestimmt die Anzahl der Markierungen pro Achse; zulässige Werte: 1 - 255
- S: legt die Skalierung (= Abstand zwischen zwei Markierungen) fest; zulässiger Bereich: -999 - 999 (Ein negatives Vorzeichen bewirkt das Zeichnen von oben nach unten bzw. von links nach rechts und ein positives Vorzeichen in umgekehrter Richtung.)
- XY: zur Bestimmung, ob X-Achse (dann XY = 1) oder Y-Achse (dann XY = 0) zu zeichnen ist

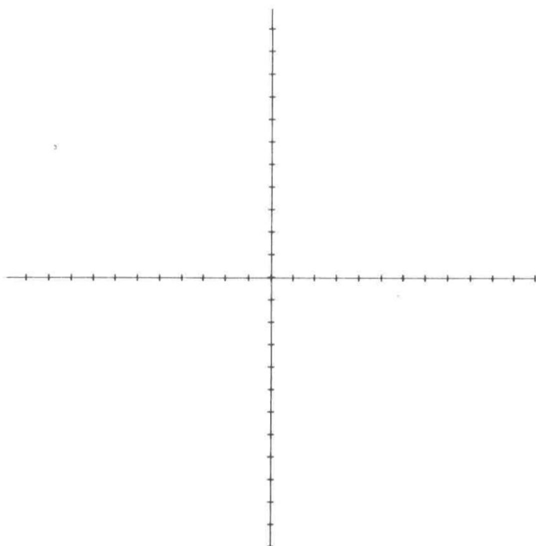
Beispiel:

10 MODE GR	Wechsel zum Grafik-Modus
20 MOVE 240, 0	Der Druckkopf wird zur Mitte der Papierbahn bewegt.
30 AXIS 0, -20, 24	zeichnet eine Y-Achse. Die erste Angabe nach dem Befehlswort AXIS (0) bestimmt, daß eine Y-Achse (Senkrechte) zu zeichnen ist. Das negative Vorzeichen (-) vor der zweiten Spezifikation bewirkt eine Zeichenrichtung von oben nach unten; die Zahl (20) bestimmt den Abstand zwischen zwei Markierungen (20 Elementarschritte = 4 mm). Die dritte Zahl (24) bestimmt die Anzahl der Markierungen.

Nach dem Zeichnen der Y-Achse steht der Schreibkopf auf der Position 240, -480. (Der X-Wert 240 resultiert aus dem Befehl der Zeile 20, der Y-Wert ergibt sich aus dem Produkt von Skalierungsfaktor (-20) und Anzahl der Markierungen (24) = -480 .

```
40 MOVE 0, -240            bewegt den Schreibkopf zur Position 0, -240
50 AXIS 1, 20, 24          zeichnet die X-Achse
60 MODE TN                 Rückkehr zum Text-Modus
```

Ergebnis eines Programmlaufs:



Zum Beschriften von Koordinatenkreuzen und anderen Zeichnungen steht - wenn man nicht zum Text-Modus zurückkehren will - folgende Anweisung zur Verfügung:

(h) GPRINT (von engl. "graphic" = grafisch und von "to print" drucken).

Es existieren zwei Formate:

Format 1: GPRINT X$

Der Befehl druckt an der aktuellen Druckkopfposition den nach dem Befehlswort angegebenen String oder den Inhalt der aufgeführten String-Variablen. Es ist meistens empfehlenswert, anschließend mit PHOME zum (Koordinaten-) Ursprung zurückzukehren.

Format 2: G P R I N T [G, L], X $ 1)

Die Anweisung druckt an der aktuellen Druckkopfposition einen String oder den Inhalt einer String-Variablen. Mit G (Bereich: 0 - 63) spezifiziert man die Größe der Zeichen und mit L (0, 1, 2 oder 3) die Lage (vgl. hierzu das folgende Beispiel).

Es ist im Regelfall ratsam, im Anschluß an die Verwendung von G P R I N T [G, L], X $ mit M O D E T N die Grundeinstellungen wieder herzustellen.

Die Wirkungsweise des Befehls ergibt sich aus dem folgenden Programm und den im Anschluß daran wiedergegebenen Ausdrucken:

Beispiel:

```
10 REM **************
20 REM * GPRINT-DEMO *
30 REM **************
40 REM LOESCHUNG ALLER ATTRIBUTE:
50 REM -------------------------
60 MODE TN
70 REM WECHSEL ZUM GRAFIK-MODUS:
80 REM -------------------------
90 MODE GR
100 REM DIV. AUSGABEN:
110 REM -------------
120 GPRINT "AUSDRUCK DER 4 LAGEN:"
130 PHOME
140 MOVE 100, -50
150 GPRINT [1, 0], "LAGE 0"
160 RMOVE 0,-20
170 GPRINT [1, 1], "LAGE 1"
180 RMOVE 0, -20
190 GPRINT [1, 2], "LAGE 2"
200 RMOVE 0, 20
210 GPRINT [1, 3], "LAGE 3"
220 MODE TN
230 MODE GR
240 MOVE 0, -200
```

1) Achtung: eckige Klammern verwenden! (vgl. auch Seite 16-19)

```
250 GPRINT "BEISPIELE FUER ZEICHENGROESS
EN:"
260 MOVE 100, -275
270 GPRINT [3, 0], "GROESSE 3"
280 MOVE 100, -350
290 GPRINT [6, 0], "GROESSE 6"
300 MOVE 100, -450
310 GPRINT [9, 0], "GR. 9"
320 REM LOESCHUNG ALLER ATTRIBUTE:
330 REM -------------------------
340 MODE TN
```

Ergebnisse eines Programmlaufs:

```
AUSDRUCK DER 4 LAGEN:

        LAGE 0

   LAGE 3      LAGE 1

        LAGE 2


BEISPIELE FUER ZEICHENGROESSEN:

        GROESSE 3
        GROESSE 6
        GR. 9
```

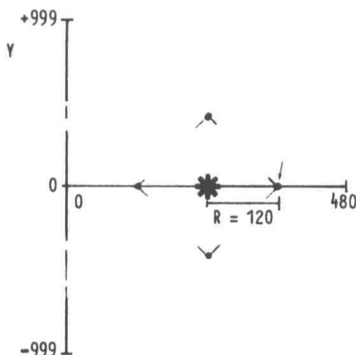
Die letzte noch zu behandelnde Anweisung für den Grafik-Modus ist

(i) CIRCLE X, Y, R, AW, EW, SW (von engl. "circle" = Kreis)

Hiermit können Vielecke und auch Kreise (genauer: extreme Vielecke) gezeichnet werden.

Die Werte im Anschluß an das Befehlswort haben folgende Bedeutung:

X, Y sind die Mittelpunktkoordinaten. Ausgehend von dem folgenden, bereits bekannten Koordinatenkreuz



hätte beispielsweise der mit einem Stern gekennzeichnete Mittelpunkt die Werte 240 (für X) und 0 (für Y).

Zur Verdeutlichung wollen wir unterstellen, daß das durch die Punkte lediglich angedeutete Viereck gezeichnet werden soll. Zu diesem Zweck ist R (= Radius) zu spezifizieren; in unserem Beispiel mit 120.

AW kennzeichnet den AnfangsWinkel (in Grad). Soll der Zeichenvorgang an der mit dem kleinen Pfeil gekennzeichneten Stelle begonnen werden, so ist AW = 0. (Wenn wir hingegen an der oberen Spitze starten wollten, so müßte man AW = 90 setzen etc.) Der Einfachheit halber wählen wir für unser Beispiel AW = 0.

EW ist der EndWinkel (in Grad); in unserem Fall muß EW = 360 betragen.

Die letzte Angabe (SW) kennzeichnet den SchrittWinkel (in Grad). Zur Erstellung eines Vierecks ist 90 anzugeben.

Der vollständige Befehl zum Zeichnen des beschriebenen Vierecks lautet also

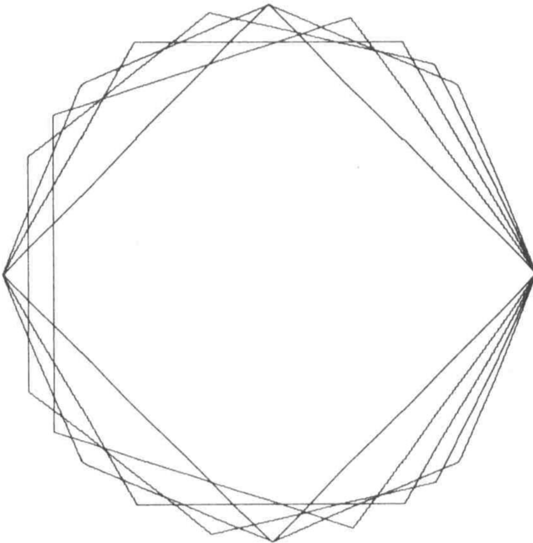
```
CIRCLE 240, 0, 120, 0, 360, 90
```

Je mehr man den Schrittwinkel reduziert, desto stärker nähert sich das Vieleck der Form eines Kreises.

Beispiel:

```
10 MODE GR
20 FOR I = 4 TO 8
30 CIRCLE 240, 0, 240, 0, 360, 360/I
40 NEXT I
50 MODE TN
```

Ergebnis eines Programmlaufs:



Einen nahezu vollkommenen Kreis bewirken wir mit SW = 0.2 - allerdings ist der Plotter-Drucker dann vergleichsweise langsam, so daß für Verarbeitungsfälle, bei denen die Geschwindigkeit wichtiger ist als die Genauigkeit, SW nur entsprechend zu erhöhen ist.

PE 16.7

(1) Entscheiden Sie, ob der jeweilige Befehl
- für Text-Modus u n d Grafik-Modus,
- n u r für Text-Modus oder
- n u r für Grafik-Modus

eingesetzt werden darf!

(Machen Sie ein Kreuz in der entsprechenden Spalte!)

Befehle	für Text-Modus u n d Grafik-Modus	n u r für Text-Modus	n u r für Grafik-Modus
TEST			
PCOLOR N			
SKIP N			
CIRCLE X,Y,R,AW,EW,SW			
PAGE N			
LIST/P			
PRINT/P			
LINE X, Y			
LINE %LT, X1, Y1, ...			
RLINE X, Y			
RLINE X1,Y1, X2,Y2,..			

Befehle	für Text-Modus und Grafik-Modus	nur für Text-Modus	nur für Grafik-Modus
GPRINT X$			
GPRINT [G, L], X$			
AXIS XY, S, Z			
RLINE %LT, X1, Y1,...			
MOVE X, Y			
RMOVE X, Y			
PHOME			
HSET			

(2) Wie lautet die Anweisung, mit der man in den Grafik-Modus wechselt?

....

(3) Wie lautet die Anweisung, mit der man in den Text-Modus zurückkehrt und gleichzeitig die Grundeinstellung der Schreibkopfposition wiederherstellt?

....

LÖS 16.7

(1)

Befehle	für Text-Modus und Grafik-Modus	nur für Text-Modus	nur für Grafik-Modus
TEST		X	
PCOLOR N	X		
SKIP N		X	
CIRCLE X,Y,R,AW,EW,SW			X
PAGE N		X	
LIST/P		X	
PRINT/P		X	
LINE X, Y			X
LINE %LT, X1, Y1,...			X
RLINE X, Y			X
RLINE X1,Y1,X2,Y2,..			X
GPRINT X$			X
GPRINT [G, L], X$			X
AXIS XY, S, Z			X
RLINE %LT,X1,Y1,...			X
MOVE X, Y			X
PHOME			X
HSET			X

(2) Wie lautet die Anweisung, mit der man in den Grafik-Modus wechselt?

Antwort: MODE GR

(3) Wie lautet die Anweisung, mit der man in den Text-Modus zurückkehrt und gleichzeitig die Grundeinstellung der Schreibkopfposition wiederherstellt?

Antwort: MODE TN

(auch richtig: MODE TL oder MODE TS ; vgl. hierzu Seite 12-11)

Sollten Ihre Lösungen nicht richtig sein, brauchen Sie die **LE 16.7** gleichwohl nicht zu wiederholen. Es ist empfehlenswerter, durch Einsatz der Befehle im Rahmen von Übungsprogrammen sich die nötige Sicherheit zu verschaffen.

ÜBUNGsprogramm zu Lektion 16

Problemstellung:

Wenn zwei einander zugeneigte Personen schriftliche Nachrichten austauschen wollen, so kann es zur Vermeidung von Peinlichkeiten sinnvoll sein, die zu übermittelnden Botschaften zu verschlüsseln.

Programmbeschreibung (Geheim-Code):

Das folgende Programm verschlüsselt einen beliebigen Text dergestalt, daß statt des jeweiligen Zeichens jenes verwendet wird, das im ASCII-Code einen um 1 höheren Wert besitzt (vgl. hierzu auch Anhang-25 f.). Das zugehörige Entschlüsselungsprogramm wird in Lektion 18 wiedergegeben.

Programmliste:

```
10 REM ***************
20 REM * GEHEIM-CODE *
30 REM ***************
40 REM
50 REM -----------------
60 REM I HAUPTPROGRAMM I
70 REM -----------------
80 GOSUB 1000 : REM TEXT-EINGABE
90 IF A$ = "XYZ" THEN 130
100 GOSUB 2000 : REM TEXT-VERSCHLUESS.
110 GOSUB 3000 : REM DRUCKER-AUSGABE
120 GOTO 80
130 END
```

```
960 REM -----------------
970 REM I UNTERPROGRAMME I
980 REM -----------------
990 REM
1000 REM TEXT-EINGABE
1010 REM ============
1020 CLS
1030 LET A$ = ""
1040 LET CT$ = ""
1050 PRINT
1060 PRINT " BITTE BELIEBIGEN TEXT EINGE
BEN!"
1070 PRINT " (ENDE-KRITERIUM: XYZ)"
1080 PRINT " NACH MAX. 40 ZEICHEN (= 1 B
ILDSCHIRM-"
1090 PRINT " ZEILE) BETAETIGEN SIE BITTE
 DIE"
1100 PRINT " 'CR'-TASTE,"
1110 PRINT " DAMIT DER VERSCHLUESSELTE T
EXT AUF DEM"
1120 PRINT " PLOTTER-DRUCKER AUSGEGEBEN
WERDEN KANN."
1130 PRINT
1140 PRINT "HIER BEGINNEN!"
1150 PRINT "↓"
1160 GET X$ : IF X$ = CHR$(13) THEN 1210
1170 IF X$ = CHR$(16) THEN LET A = LEN(A
$) : LET A$ = LEFT$(A$, A-1) : PRINT CHR
$(16); : GOTO 1160
1180 PRINT X$;
1190 LET A$ = A$ + X$
1200 GOTO 1160
1210 RETURN
2000 REM TEXT-VERSCHLUESSELUNG
2010 REM =====================
2020 PRINT : PRINT : PRINT
2030 PRINT " VERSCHLUESSELTER TEXT:"
2040 PRINT " :::::::::::::::::::::::"
2050 PRINT
2060 FOR I = 1 TO LEN(A$)
2070 LET B$ = MID$(A$, I, 1)
2080 LET C = ASC(B$) + 1
2090 LET CT$ = CT$ + CHR$(C)
2100 NEXT I
```

```
2110 PRINT CT$
2120 RETURN
3000 REM DRUCKER-AUSGABE
3010 REM ===============
3020 FOR I = 1 TO 4 : PRINT : NEXT I
3030 PRINT " AUSGABE AUF PLOTTER-DRUCKER
 (J/N) ?"
3040 GET JN$
3050 IF JN$ = "J" THEN 3080
3060 IF JN$ = "N" THEN 3510
3070 GOTO 3040
3080 REM WAHL DER DRUCK-FARBE
3090 REM --------------------
3100 CLS
3110 PRINT
3120 PRINT " WELCHE FARBE ? "
3130 PRINT : PRINT : PRINT
3140 PRINT " SCHWARZ        (1)"
3150 PRINT
3160 PRINT " BLAU           (2)"
3170 PRINT
3180 PRINT " GRUEN          (3)"
3190 PRINT
3200 PRINT " ROT            (4)"
3210 PRINT
3220 FOR I = 1 TO 10 : PRINT : NEXT I
3230 PRINT " '1' OD. '2' OD. '3' OD. '4'
 DRUECKEN!"
3240 GET DF$
3250 IF (DF$="1") + (DF$="2") + (DF$="3"
) + (DF$="4") THEN 3270
3260 GOTO 3240
3270 PCOLOR VAL(DF$) - 1
3280 REM WAHL DER SCHRIFTGROESSE
3290 REM -----------------------
3300 CLS
3310 PRINT
3320 PRINT " WELCHE SCHRIFTGROESSE ?"
3330 PRINT : PRINT : PRINT : PRINT
3340 PRINT " KLEIN          (1)"
3350 PRINT : PRINT
3360 PRINT " NORMAL         (2)"
3370 PRINT : PRINT
3380 PRINT " GROSS          (3)"
```

```
3390 FOR I = 1 TO 10 : PRINT : NEXT I
3400 PRINT " '1' OD. '2' OD. '3' DRUECKE
N!"
3410 GET SG$
3420 ON VAL(SG$) GOTO 3440, 3450, 3460
3430 GOTO 3410
3440 MODE TS : GOTO 3480
3450 MODE TN : GOTO 3480
3460 MODE TL : GOTO 3480
3470 REM
3480 REM AUSGABE
3490 REM -------
3500 PRINT/P CT$
3510 RETURN
```

E r l ä u t e r u n g e n:

Auf den ersten Blick unverständlich ist die Eingabeprozedur der Zeilen 1160 - 1200. Wir mußten uns für diese etwas umständliche Befehlsfolge entscheiden, weil es mit I N P U T nicht möglich ist, ein Komma (als Teil des Strings) einzugeben; denn wenn man im Rahmen einer INPUT-Eingabe ein Komma setzt, unterstellt das System, daß die so getrennten Zeichen in verschiedenen Variablen abzulegen sind.

Die Anweisungen der Zeilen 1160 und 1200 bilden Anfang und Ende einer Schleife zur Eingabe jeweils eines Zeichens nach X$, das dann in Zeile 1180 (PRINT X$;) ausgegeben und anschließend in Zeile 1190 an das vorangehende angehängt (konkateniert) wird. Wenn aufgrund der Abfrage in Zeile 1160 (IF X$ = ...) erkannt wird, daß die CR -Taste (≙ CHR$(13)) gedrückt wurde, verzweigt das Programm zur Zeile 1210 (RETURN), womit die Eingabe und die Ausführung des Unterprogramms abgeschlossen sind.

Da bei Betätigung der DEL -Taste nicht etwa ein Zeichen gelöscht wird, sondern der entsprechende ASCII-Wert (dez. 16) nach X$ gelangt, überprüfen wir in Zeile 1170, ob X$ = CHR$(16) ist. Im Ja-Fall ermitteln wir die Länge von A$ (LET A = LEN(A$)), verkürzen A$ um das letzte, hinzugefügte Zeichen (LET A$ = LEFT$(A$, A-1)), tilgen auf dem Bildschirm das letzte Zeichen (PRINT CHR$(16);), um dann zur Zeile 1160 zurückzukehren.

Die eigentliche Verschlüsselung erfolgt in der FOR... NEXT-Schleife der Zeilen 2060 - 2100. Die Schleifenbefehle werden entsprechend der Länge des in A$ befindlichen Strings (FOR I = 1 TO LEN(A$)) wiederholt. In Zeile 2070 weisen wir B$ zunächst das erste Zeichen aus A$ (I = 1) zu; in Zeile 2080 addieren wir zum ASCII-Wert dieses Zeichens den Wert 1 (ASC(B$) + 1)) und weisen diesen Wert der Variablen C zu. In Zeile 2090 ermitteln wir für den Inhalt von C das entsprechende ASCII-Zeichen und konkatenieren es in CT$. Diese Prozedur wird anschließend mit dem jeweils nächsten Zeichen aus A$ vollzogen, und zwar so lange, bis alle Zeichen in CT$ in verschlüsselter Form abgelegt sind.

Die Ausgabe von CT$ auf den Bildschirm erfolgt in Zeile 2110 und ggf. auf den Plotter-Drucker in Zeile 3500.

17. Musik und Geräusche

LE 17.1

Neben den zahlreichen grafischen und farblichen Gestaltungsmöglichkeiten gestattet der SHARP MZ-700 auch die musikalische Untermalung der Ausgaben.

Für jene, die sich wenig mit Musik beschäftigt haben, sei folgendes in Erinnerung gerufen:

Die Stammtöne einer C-Dur-Tonleiter heißen

C, D, E, F, G, A, H, (C) .

Im Englischen (und deshalb auch in BASIC) bezeichnet man das H mit B [1] - alle anderen Töne heißen wie im Deutschen:

C, D, E, F, G, A, B .

↑ Hier besteht eine Abweichung gegenüber der deutschen Bezeichnung.

(Ab jetzt verwenden wir nur noch die englischen Bezeichnungen.)

Mit Hilfe der Anweisung

MUSIC A$ (von engl. "music" = Musik)

kann man nun jene Töne erklingen lassen, die im Anschluß an das

1) Die Situation ist hier ein klein wenig verwirrend: Der "deutsche" Ton H heißt heißt im Englischen B; der "deutsche" Ton Be (= das um einen Halbton verminderte H) heißt im Englischen B/flat.

Befehlswort als String oder mit einer String-Variablen spezifiziert sind.

Beispiel:

```
10 MUSIC "C, D, E, F, G, A, B"
```

Insgesamt steht ein Tonumfang von drei Oktaven zur Verfügung, nämlich vom unteren (kleinen) C bis zum oberen (zweigestrichenen) B.

Sollen die Töne der unteren Oktave erklingen, sind dem Notennamen je ein Minuszeichen (-) voranzustellen; für die mittlere Oktave entfällt ein Spezifikationszeichen; für die Töne der oberen Oktave verwendet man das Pluszeichen (+).

Beispiel:

```
10 REM UNTERE OKTAVE
20 REM -------------
30 MUSIC "-C,-D,-E,-F,-G,-A,-B"
40 REM MITTLERE OKTAVE
50 REM ---------------
60 MUSIC " C, D, E, F, G, A, B"
70 REM OBERE OKTAVE
80 REM ------------
90 MUSIC "+C,+D,+E,+F,+G,+A,+B"
```

PE 17.1

(Bitte, die linke Seite abdecken!)

Welche der folgenden Behauptungen ist/sind richtig?

A Mit Hilfe des Befehls M U R K S A $ kann man Töne erklingen lassen.

B Mit Hilfe der Anweisung M U S I C A $ kann man jene Töne erklingen lassen, die im Anschluß an das Befehlswort als String oder mit einer String-Variablen spezifiziert sind.

C Beim SHARP MZ-700 steht nur ein einziger Ton zur Verfügung, weil Computer letztlich unmusikalisch sind.

D Beim SHARP MZ-700 steht ein Tonumfang von genau einer Oktave zur Verfügung.

E Beim SHARP MZ-700 steht ein Tonumfang von drei Oktaven zur Verfügung, nämlich vom unteren (kleinen) C bis zum oberen (zweigestrichenen) B (engl. Bez.).

F Beim MUSIC-Befehl sind die Töne in Anlehnung an die italienischen (internationalen) Bezeichnungen (do, re, mi, fa, so, la, ti) zu spezifizieren.

G Beim MUSIC-Befehl sind die Töne in Anlehnung an die deutschen Bezeichnungen (C, D, E, F, G, A, H) zu spezifizieren - gewissermaßen als Verneigung vor dem großen Sänger Heino Gendrich.

H Beim MUSIC-Befehl sind die Töne in Anlehnung an die englischen Bezeichnungen (C, D, E, F, G, A, B) zu spezifizieren.

Der/die Lösungsbuchstabe/n lautet/lauten

B, E, H ....

LÖS 17.1

Die Lösungsbuchstaben lauten

B, E, H .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 17.1** !

LE 17.2

Neben den Tönen

-C, -D, -E, -F, -G, -A, -B, C, D, E, F, G, A, B,
+C, +D, +E, +F, +G, +A, +B

gibt es

(1) Zwischentöne,

die dadurch erzeugt werden, daß man der Tonbezeichnung ein # voranstellt, wodurch sich die Tonhöhe um einen halben Wert erhöht - eine Darstellungsform, wie sie auch in der Notenliteratur üblich ist.

Beispiel:

```
10 LET A$ = "+A, +#A"
20 FOR I = 1 TO 5
30 MUSIC A$
40 NEXT I
```

(2) Tondauer

Durch Anhängen einer ganzen Zahl von 0 bis 9 an die Tonbezeichnung kann man die Tondauer festlegen. Die Zahl 0 bedeutet 1/32, die Zahl 9 eine ganze Note (vgl. hierzu auch die Aufstellung auf der folgenden Seite). Eine einmal erfolgte Tondauer-Spezifikation gilt so lange, bis eine neue Angabe erfolgt. Ohne jegliche Tondauer-Bestimmung ertönen 1/32 Noten.

(3) Pausen

Eine Pause bewirkt man mit dem Zeichen

R (von engl. "rest" = Pause).

Ihre Länge läßt sich in analoger Weise zur Tondauer variieren.

Zusammenfassung:

Pausen:



Tondauer:



1/32	1/16	gepunkt. 1/16	1/8	gepunkt. 1/8	1/4	gepunkt. 1/4	1/2	gepunkt. 1/2	ganze

(Pause bzw. Note)

Spezifikation im Rahmen des MUSIC-Befehls:

0	1	2	3	4	5	6	7	8	9

Aufbau einer Note im Rahmen des MUSIC-Befehls:

Oktavenbezeichn. / Halbtonspez. / Tonbezeichnung / Tondauer

<u>Hinweis:</u> Folgen zweimal dieselben Töne in einem MUSIC-Befehl unmittelbar aufeinander, so sind sie nicht als zwei Töne erkennbar. Beispielsweise ist die Wirkung von

```
MUSIC "C7"
```

und

```
MUSIC "C5, C"
```

vollkommen identisch.

Wenn "C5, C" tatsächlich als zwei getrennte Töne wahrgenommen werden sollen, so gibt es hierfür zwei Realisierungsmöglichkeiten:

(1) Die Töne sind durch zwei getrennte MUSIC-Anweisungen aufzurufen:

```
.. MUSIC "C5"
.. MUSIC "C"
```

(2) Die Tondauer des ersten Tons ist etwas zu verringern und eine kleine Pause einzufügen:

```
.. MUSIC "C4, R2, C5"
```

<u>Beispiel:</u>

```
10 REM ******************
20 REM * KUKUCK, KUKUCK *
30 REM ******************
40 LET A1$ = "+C5,A,R" : REM  1.TAKT
50 LET A2$ = "+C,A,R"  : REM  2.TAKT
60 LET A3$ = "G,F,G"   : REM  3.TAKT
70 LET A4$ = "F7,R5"   : REM  4.TAKT
80 LET A5$ = "G"       : REM  5.TAKT
90 LET A6$ = "G,A"
100 LET A7$ = "#A7,G5" : REM  6.TAKT
110 LET A8$ = "A"      : REM  7.TAKT
120 LET A9$ = "A,#A"
130 LET B1$ = "+C7,A5" : REM  8.TAKT
140 LET B2$ = "+C7,A5" : REM  9.TAKT
150 LET B3$ = "+C7,A5" : REM 10.TAKT
160 LET B4$ = "#A,A,G" : REM 11.TAKT
170 LET B5$ = "F7"     : REM 12.TAKT
```

```
180 MUSIC A1$,A2$,A3$,A4$,A5$
190 MUSIC A6$,A7$,A8$
200 MUSIC A9$,B1$,B2$,B3$,B4$,B5$
210 END
```

Das aufgeführte Kinderlied wurde zwar in Anlehnung an die Noten umgesetzt; die Melodie ertönt aber zu langsam.

Ein solcher Mangel läßt sich mit der Anweisung

TEMPO X (von engl. "tempo" = Geschwindigkeit)

beheben, wobei X einen Wert von 1 bis 7 annehmen darf:

Beispiele:

	Bedeutung:
TEMPO 1	langsamstes Tempo (lento, adagio)
TEMPO 4	mittleres Tempo (moderato)
TEMPO 7	schnellstes Tempo (molto allegro, presto)

Ohne TEMPO-Befehl arbeitet das System mit TEMPO 4; in unserem obigen Programm sollten wir folgende Instruktion ergänzen:

```
35 TEMPO 6
```

Für das Umsetzen von Melodien auf den SHARP MZ-700 wird im Regelfall eine Notenvorschrift die Ausgangsbasis zu bilden haben (vgl. hierzu **ÜBUNG**sprogramm zu Lektion 17). Den Noten ordnet man dann die Tonbezeichnungen (ggf. mit vorangestellter Oktavenangabe / evtl. auch mit einer Halbton-Spezifikation) und die nachgestellte Tondauer (nur wenn sie sich geändert hat!) zu. Außerdem sind die Pausen in Anlehnung an die Aufstellung auf Seite 17-06 festzulegen.

"Geräusche" erzeugt man durch die Kombination von Extremwerten hinsichtlich Tonhöhe und / oder Tondauer, für die dann meistens eine mehrmalige Wiederholung vorzusehen ist.

Beispiel:

```
10 REM *******************************
20 REM * SIRENE EINES POLIZEI-AUTOS *
30 REM *******************************
40 TEMPO 4
50 FOR I = 1 TO 10
60 MUSIC "-A1, A, -A0, A5"
70 NEXT I
```

PE 17.2

(1) Wie schreibt man die folgenden Noten im Rahmen des MUSIC-Befehls?

(Für die Lösung der Aufgaben dürfen Sie die Zusammenfassung auf Seite 17-06 verwenden!)

(a) Ton A der unteren Oktave als 1/2 Note:

....A7....................

(b) Ton A der mittleren Oktave als 1/4 Note:

....A5....................

(c) Ton G der oberen Oktave als gepunktete 1/8 Note:

...+G.4...............

(d) Ton Gis (= um einen Halbton erhöhtes G) der oberen Oktave als 1/16 Note:

..+#G1....................

(2) Wie spezifiziert man im Rahmen eines MUSIC-Befehls eine Pause, die der Länge einer 1/2 Note entspricht?

...R.7................

(3) Eine für den SHARP MZ-700 programmierte Melodie ist zu langsam und soll durch Einfügung einer Anweisung auf das schnellst mögliche Tempo gesteigert werden. Wie lautet die Instruktion?

35 . . Tempo 7 . . . .

LÖS 17.2

(1) Die aufgeführten Noten haben im Rahmen des MUSIC-Befehls das folgende Aussehen:

(a) Ton A der unteren Oktave als 1/2 Note:

"-A7"

(b) Ton A der mittleren Oktave als 1/4 Note:

"A5"

(c) Ton G der oberen Oktave als gepunktete 1/8 Note:

"+G4"

(d) Ton Gis (= um einen Halbton erhöhtes G) der oberen Oktave als 1/16 Note:

"+#G1"

(2) Eine Pause, die der Länge einer 1/2 Note entspricht, spezifiziert man im Rahmen eines MUSIC-Befehls folgendermaßen:

"R7"

(3) Eine für den SHARP MZ-700 programmierte Melodie ist zu langsam und soll durch Einfügung einer Anweisung auf das schnellst mögliche Tempo gesteigert werden. Die Instruktion lautet:

35 TEMPO 7

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 17.1** !

ÜBUNGsprogramm zu Lektion 17

Programmbeschreibung:

Zum Zwecke des Mitsingens soll der amerikanische folk-song "Michael, row the boat ashore" programmiert werden. (Das Lied stammt von den Kücsteninseln von Georgia, wo früher die Waren der ankommenden Frachtschiffe mit langen, schmalen Booten zum Festland gebracht wurden. Der Name Michael bezieht sich auf den Erzengel Michael.)

Noten mit den zugeordneten String-Werten:

Mich- ael row the boat a- shore, ha- le- lu- u- u-

R7 D5 #F A6 #F3 A5 B A7 #F5 A B7 B3 A3 B5

ja, Mich-ael, row the boat a- shore ha- le- lu- u- ja!

A7 #F5 A A6 #F3 G5 #F E7 D5 E #F7 E D8 R5

2. Sister help to trim the sail, haleluja
   Sister help to trim the sail, haleluja

3. Jordan's River is deep and wide, haleluja
   Meet my lover on the other side, haleluja

4. Jordan's River is chilly and cold, haleluja
   Kills the body but not the soul, haleluja

Programmliste:

```
10 REM ******************************
20 REM * MICHAEL, ROW THE BOAT ASHORE *
30 REM ******************************
40 TEMPO 4
50 LET P1$ = "R7"
60 LET A1$ = "D5, #F"
70 LET A2$ = "A6, #F3, A5, B"
80 LET A3$ = "A7, #F5, A"
90 LET A4$ = "B7, B3, A3, B5"
100 LET A5$ = "A7, #F5, A"
110 LET A6$ = "A6, #F3, G5, #F"
120 LET A7$ = "E7, D5, E"
130 LET A8$ = "#F7, E"
140 LET A9$ = "D8"
150 LET P2$ = "R5"
160 FOR I = 1 TO 4
170 MUSIC P1$
180 MUSIC A1$, A2$, A3$, A4$
190 MUSIC A5$, A6$, A7$, A8$, A9$
200 MUSIC P2$
210 NEXT I
220 END
```

18. Springen mit GOTO und Verzweigungen mit IF ... THEN

Vergleichsoperatoren und logische Operatoren

LE 18.1

Wir haben den bedingungsfreien Sprung mit GOTO und die bedingte Verzweigung mit IF ... THEN in vorangegangenen Lektionen schon verwenden müssen, obwohl die Befehle noch nicht eingeführt waren. Gleichwohl meinen wir, daß man die Wirkungsweise der Anweisungen aus dem Zusammenhang heraus verstehen konnte. Die genauen Erläuterungen wollen wir jetzt nachholen.

Mit dem Befehl

GOTO (von engl. "to go to" = gehen nach)

können wir von jeder beliebigen Stelle eines Programms zu jeder beliebigen anderen Stelle dieses Programms springen. Das Sprungziel wird durch die Zeilennummer festgelegt, die dem Operationsteil (GOTO[1]) folgt.

Beispiel:

```
10 REM SPRINGEN MIT GOTO
20 REM -----------------
30 PRINT "DER GOTO-BEFEHL WURDE"
40 GOTO 80
50 PRINT "WENN DIESER TEXT AUF DEM"
60 PRINT "BILDSCHIRM AUSGEGEBEN WIRD,"
70 PRINT "DANN LIEGT EIN FEHLER VOR!"
80 PRINT "KORREKT AUSGEFUEHRT!"
90 END
```

1) Beim SHARP MZ-700 ist es auch zulässig, in Anlehnung an die im Englischen übliche Schreibweise "GO" und "TO" getrennt einzugeben, also .. GO TO .. . Nach erneutem Auflisten erscheint das Befehlswort aber ohne Leerstelle.

IF ... THEN (von engl. "if ... then" = falls ... dann)

Häufig ist es erforderlich, einen Sprung im Programm lediglich dann auszuführen, wenn eine bestimmte Bedingung erfüllt ist. Beispielsweise soll ein Sprung ausschließlich erfolgen, wenn eine numerische Variable einen konkreten Wert überschreitet. So sind etwa in einem Lohnprogramm die Befehle zur Berechnung der Lohnsteuer nur anzuspringen, wenn der Lohn eine festgelegte Höhe übersteigt (Minimalbeträge sind lohnsteuerfrei). Das Überschreiten einer bestimmten Lohnhöhe wäre die zu überprüfende Bedingung.

Für bedingte Verzweigungen stehen uns verschiedene IF-Anweisungen zur Verfügung, wobei man die eigentlichen Bedingungen mit Hilfe der folgenden Vergleichsoperatoren formuliert.

Vergleichsoperatoren	Bedeutung	
=	gleich	
<> (auch zulässig: ><)	ungleich	
<	kleiner als	1)
>	größer als	1)
<= (auch zulässig: =<)	kleiner als oder gleich	
>= (auch zulässig: =>)	größer als oder gleich	

1) Mnemotechnischer Hinweis: Die Vergleichsoperatoren für "kleiner als" (<) und für "größer als" (>) werden bisweilen verwechselt. Dem ist mit einer kleinen "Eselsbrücke" abzuhelfen: Stellt man dem "Kleiner-als-Operator" einen senkrechten Strich voran, also |<, dann erkennt man den Anfangsbuchstaben des Operators (k leiner als). Umgekehrt läßt sich aus dem "Größer-als-Operator" (>) mit etwas Phantasie ein "G" bilden: >.

Format 1: IF ... THEN Zeilennummer

Beispiele:

(1)

```
 .
 .
100 IF A > 9 THEN 200
110 PRINT "A IST NICHT GROESSER ALS 9!"
 .
 .
```

Erläuterungen: Falls die Variable A größer als 9 ist, dann wird zur Befehlszeile 200 gesprungen; andernfalls setzt der Programmlauf mit der unmittelbar folgenden Befehlszeile 110 fort.

(2)

```
 .
 .
30 IF X * P > Z↑2 THEN 90
40 PRINT "BEDINGUNG NICHT ERFUELLT!"
 .
 .
```

Erläuterungen: Falls das Ergebnis aus X * P größer ist als das Quadrat von Z, dann wird zur Befehlszeile 90 gesprungen; andernfalls setzt der Programmlauf mit der unmittelbar folgenden Befehlszeile 40 fort.

(Vor dem eigentlichen Vergleich erfolgt also die Berechnung der mathematischen Ausdrücke, d. h. die Ausführung der Vergleichsoperatoren erfolgt nach den arithmetischen Operatoren.)

(3)

```
 .
 .
100 IF A$ = "L" THEN 250
110 IF A$ = "$" THEN 400
120 IF A$ = "DM" THEN 550
 .
 .
```

Erläuterungen: Falls der Inhalt von A$ der String "L" ("L" z. B. für "Lire") ist, dann wird zur Befehlszeile 250 verzweigt; andernfalls setzt der Programmlauf mit der unmittelbar folgenden Befehlszeile 110 fort. Sie enthält die Bedingungsabfrage, ob der Inhalt von A$ der String "$" ist. Fall dies zutrifft, wird zur Zeile 400 verzweigt; andernfalls setzt der Programmlauf mit der unmittelbar folgenden Befehlszeile 120 fort, die eine weitere Bedingungsabfrage enthält.

PE 18.1

Ergänzen Sie auf den Seiten 18-07 / 18-08 die fehlenden BASIC-Befehle zu der folgenden Aufgabe!

P R O G R A M M B E I S P I E L

A u f g a b e n d e f i n i t i o n :

Für Ihre Urlaubsplanung (und ggf. für die Ihrer Freunde) soll Ihnen der SHARP MZ-700 errechnen, wieviel Devisen in Lire oder Pfund oder Francs Sie für das ersparte Geld (DM-Betrag) erhalten.

Folgende Variablennamen sind zu verwenden:

DM = erspartes Geld

A$ = gewünschte Währung

Zu Beginn des Programms werden den folgenden String-Variablen die aufgeführten Zeichenkettenkonstanten zugeordnet:

K0$ = "DM-BETRAG"

K1$ = "WELCHE WAEHRUNG?"

K2$ = "LIRE / PFUND / FRANCS"

K3$ = "UNZULAESSIGE WAEHRUNG!"

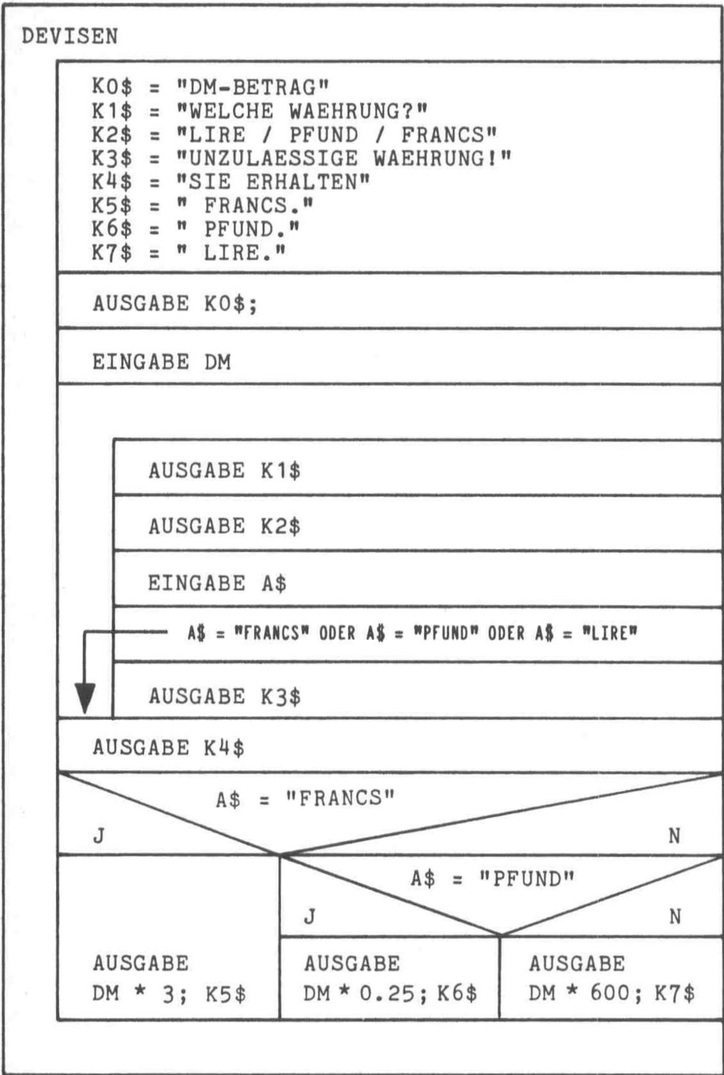
K4$ = "SIE ERHALTEN"

K5$ = " FRANCS."

K6$ = " PFUND."

K7$ = " LIRE."

Struktogramm:



Hinweise:

(1) Im nebenstehenden Struktogramm sind jene Befehle nicht berücksichtigt, die nur der optischen Verfeinerung dienen, also "Bildschirm löschen" (CLS) und "Leerzeile" (PRINT).

(2) Die den Umrechnungen zugrunde liegenden Wechselkurse entsprechen nicht den realen Kursen eines bestimmten Tages.

Ergänzen Sie die fehlenden und unvollständigen Befehle zum nebenstehenden Struktogramm!

Programmliste:

```
10 REM ***********
20 REM * DEVISEN *
30 REM ***********
40 LET K0$ = "DM-BETRAG"
50 LET K1$ = "WELCHE WAEHRUNG?"
60 LET K2$ = "LIRE / PFUND / FRANCS"
70 LET K3$ = "UNZULAESSIGE WAEHRUNG!"
80 LET K4$ = "SIE ERHALTEN"
90 LET K5$ = " FRANCS."
100 LET K6$ = " PFUND."
110 LET K7$ = " LIRE."
120 CLS
130 PRINT K0$;

140 ..............................     <------ Ergänzen!
150 REM BEGINN DER M-SCHLEIFE
160 REM --------------------
170 PRINT
180 PRINT K1$
190 PRINT
```

```
200 PRINT K2$
210 PRINT

220 ..............................     <------ Ergänzen!
230 PRINT
240 IF A$ = "FRANCS" THEN 290
250 IF A$ = "PFUND" THEN 290
260 IF A$ = "LIRE" THEN 290
270 PRINT K3$

280 GOTO .....................     <------ Ergänzen!
290 REM ENDE DER M-SCHLEIFE
300 REM -------------------
310 PRINT K4$
320 PRINT
330 IF A$ = "FRANCS" THEN 390

340 ..............................     <------ Ergänzen!
350 REM AUSGABE LIRE
360 REM -----------
370 PRINT DM * 600; K7$
380 GOTO 460
390 REM AUSGABE FRANCS
400 REM -------------
410 PRINT DM * 3; K5$

420 ..............................     <------ Ergänzen!
430 REM AUSGABE PFUND
440 REM ------------
450 PRINT DM * .25; K6$
460 PRINT
470 END
```

Die in diesen drei Befehlen enthaltenen Einzelbedingungen lassen sich auch zu einer Mehrfachbedingung zusammenfassen. (Vgl. hierzu die Ausführungen auf Seite 18-20 ff.)

LÖS 18.1

Programm der PE 18.1 mit den Ergänzungen:

```
10 REM ***********
20 REM * DEVISEN *
30 REM ***********
40 LET K0$ = "DM-BETRAG"
50 LET K1$ = "WELCHE WAEHRUNG?"
60 LET K2$ = "LIRE / PFUND / FRANCS"
70 LET K3$ = "UNZULAESSIGE WAEHRUNG!"
80 LET K4$ = "SIE ERHALTEN"
90 LET K5$ = " FRANCS."
100 LET K6$ = " PFUND."
110 LET K7$ = " LIRE."
120 CLS
130 PRINT K0$;
140 INPUT DM
```



```
150 REM BEGINN DER M-SCHLEIFE
160 REM ---------------------
170 PRINT
180 PRINT K1$
190 PRINT
200 PRINT K2$
210 PRINT
220 INPUT A$
```



```
230 PRINT
240 IF A$ = "FRANCS" THEN 290
250 IF A$ = "PFUND" THEN 290
260 IF A$ = "LIRE" THEN 290
```

```
270 PRINT K3$
280 GOTO 150
```



```
290 REM ENDE DER M-SCHLEIFE
300 REM ------------------
310 PRINT K4$
320 PRINT
330 IF A$ = "FRANCS" THEN 390
340 IF A$ = "PFUND" THEN 430
```



```
350 REM AUSGABE LIRE
360 REM -----------
370 PRINT DM * 600; K7$
380 GOTO 460
390 REM AUSGABE FRANCS
400 REM -------------
410 PRINT DM * 3; K5$
420 GOTO 460
```



```
430 REM AUSGABE PFUND
440 REM ------------
450 PRINT DM * .25; K6$
460 PRINT
470 END
```

Sollten Ihre Ergänzungen nicht richtig sein, kehren Sie zurück zur → **LE 18.1** und wiederholen Sie ggf. auch **LE 15.1** !

LE 18.2

Die IF... THEN-Anweisung (Format 1), die wir in der vorangegangenen **LE 18.1** vorgestellt haben, steht so bei allen bekannten BASIC-Rechnern zur Verfügung. Zur Verdeutlichung soll die Funktionsweise des Befehls noch einmal kurz beschrieben werden:

Nach IF folgt die zu überprüfende Bedingung. Wenn sie erfüllt ist, wird zu der nach THEN angegebenen Zeilennummer gesprungen. Ist die Bedingung hingegen nicht erfüllt, dann wird die Programmausführung bei dem / den Befehl(en) der unmittelbar folgenden Zeilennummer fortgesetzt. Anders ausgedrückt: Mit einem IF ... THEN -Befehl des Formats 1 bewirkt man eine Entscheidung dahingehend, ob ein Sprung im Programm auszuführen ist oder nicht.

Beispiel (Bedingung ist nicht erfüllt):
(Variable A habe den Wert 10)

```
.
.
100 IF A = 20 THEN 200
110 LET A = A + 1
.
.
```

Nach Ausführung des Befehls in Zeile 100 wird der Programmlauf bei Zeile 110 fortgesetzt.

Beispiel (Bedingung ist erfüllt):
(Variable A habe den Wert 20)

```
 .
 .
100 IF A = 20 THEN 200 ──┐
110 LET A = A + 1        │
 .                       │
 .                       │
┌────────────────────────┘
└►200 LET A = 0
 .
 .
```

Nach Ausführung des Befehls in Zeile 100 wird zum Befehl in Zeile 200 gesprungen.

Neben dem beschriebenen Format 1 der IF... THEN-Anweisung gibt es für diesen Befehl weitere Möglichkeiten.

Format 2: IF ... THEN Befehl(e)

Beispiel:

```
 .
 .
60 LET A = 8.5
 .
 .
100 IF A = 20 THEN LET A = 0
110 LET A = A + 1
 .
 .
```

Bei der IF...THEN-Anweisung des Formats 2 steht (stehen) nach THEN statt einer Zeilennummer ein (oder mehrere, durch Doppelpunkte zu trennende) Befehl(e). Wenn die Bedingung erfüllt ist, wird (werden) zunächst der (die) auf THEN folgende(n) Befehl(e)

ausgeführt. Ist sie hingegen nicht erfüllt, dann setzt der Programmlauf mit dem Befehl der unmittelbar folgenden Zeilennummer fort. Anders ausgedrückt: Mit einer IF ... THEN-Anweisung des Formats 2 erfolgt eine Entscheidung darüber, ob der auf THEN folgende Befehl (im Beispiel: LET A = 0) ausgeführt wird oder nicht.

Format 3: IF ... GOTO Zeilennummer

Beispiel:

```
.
.
50 IF A < 20 GOTO 90
60 LET C = D * B
.
.
```

In stärkerer Anlehnung an die englische Umgangssprache ist es beim SHARP MZ-700 auch zulässig, nach der Bedingungsabfrage (IF A < 20) den (bedingt) auszuführenden Sprung nur mit GOTO (also ohne THEN) zu formulieren.

Ergänzender Hinweis für den fortgeschrittenen Leser:

(Der Anfänger sollte die folgenden Ausführungen übergehen!)

Das Ineinanderschachteln von IF ... THEN-Anweisungen ist zulässig. Eine Beschränkung ergibt sich nur aus der maximal zulässigen Länge einer Befehlszeile.

Beispiel: .. IF A = B THEN IF B = C THEN PRINT "A = C"

PE 18.2

Ergänzen Sie die Erläuterungen zu den aufgeführten Beispielen!

Beispiel (1):

```
  .
  .
100 IF HA > 40 THEN 700
110 LET E = SQR(P)
  .
  .
```

Erläuterungen:

(a) Wenn HA = 10 ist, dann wird nach dem Befehl in Zeile 100 der Programmlauf in der folgenden Zeile fortgesetzt:

. . . .

(b) Wenn HA = 40 ist, dann wird nach dem Befehl in Zeile 100 der Programmlauf in der folgenden Zeile fortgesetzt:

. . . .

(c) Wenn HA = 90 ist, dann wird nach dem Befehl in Zeile 100 der Programmlauf in der folgenden Zeile forgesetzt:

. . . .

Beispiel (2):

```
    .
    .
200 IF MA < 20 THEN LET MA = 0
210 LET MA = MA + 1
    .
    .
```

Erläuterungen:

(a) Wenn MA = 10 ist, dann hat nach Ausführung des Befehls in Zeile 210 die Variable MA folgenden Inhalt:

....

(b) Wenn MA = 20 ist, dann hat nach Ausführung des Befehls in Zeile 210 die Variable MA folgenden Inhalt:

....

(c) Wenn MA = 30 ist, dann hat nach Ausführung des Befehls in Zeile 210 die Variable MA folgenden Inhalt:

....

Beispiel (3):

```
.
.
100 IF KA < 100 GOTO 300
110 LET P = C * F
.
.
```

Erläuterungen:

(a) Wenn KA = 10 ist, dann wird nach dem Befehl in Zeile 100 der Programmlauf in der folgenden Zeile fortgesetzt:

. . . .

(b) Wenn KA = 100 ist, dann wird nach dem Befehl in Zeile 100 der Programmlauf in der folgenden Zeile fortgesetzt:

. . . .

(c) Wenn KA = 200 ist, dann wird nach dem Befehl in Zeile 100 der Programmlauf in der folgenden Zeile fortgesetzt:

. . . .

LÖS 18.2

Beispiel (1):

	Angenommene Werte der Variablen HA	Fortsetzung des Programmlaufs in Zeile
(a)	10	110
(b)	40	110
(c)	90	700

Beispiel (2):

	Angenommene Werte der Variablen MA	Werte der Variablen MA nach Ausführung des Befehls in Zeile 210
(a)	10	1
(b)	20	21
(c)	30	31

Beispiel (3):

	Angenommene Werte der Variablen KA	Fortsetzung des Programmlaufs in Zeile
(a)	10	300
(b)	100	110
(c)	200	110

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 18.1** !

LE 18.3

Eine Bedingung ist entweder erfüllt oder nicht erfüllt. In der Fachsprache sagt man auch, eine Bedingung sei "wahr" (= erfüllt) oder "falsch" (= nicht erfüllt). Statt des Wortes "wahr" verwendet der SHARP MZ-700 dafür den Ziffernwert -1 und statt "falsch" den Ziffernwert 0.

Beispiele:

(a)

```
10 LET A = 5
20 IF A = 5 THEN 200
 .
 .
```

Diese Bedingung ist erfüllt, also "wahr". Demnach hat sie den Wert -1.

(b)

```
10 LET B = 6
20 IF B = 20 THEN 200
 .
 .
```

Diese Bedingung ist nicht erfüllt, also "falsch". Demnach hat sie den Wert 0.

---

1) Um zu ermitteln, welchen Wert ein System für "wahr" oder "falsch" verwendet, kann man sich diesen in BASIC mit PRINT ausgeben lassen.

Beispiel:

```
10 LET A = 10
20 PRINT A < 20
30 PRINT A > 20
RUN
-1
 0
Ready
```

Bei den meisten BASIC-Systemen - so auch beim SHARP MZ-700 - ist es zusätzlich möglich, zusammengesetzte Bedingungen zu formulieren. Hierfür stehen sog. logische Operatoren zur Verfügung, und zwar vor allem für das "logische UND" und für das "logische inklusive ODER"[1]. Eine derart kombinierte Bedingung hat dann den Wert <0 für "wahr" (= erfüllt) und 0 für "falsch" (= nicht erfüllt).

Allerdings weichen die logischen Operatoren in der hier vorgestellten BASIC-Version von der sonst üblichen Schreibweise[2] ab.

Logisches UND: *

Die mit dem logischen Operator * zusammengesetzte Bedingung ist dann "wahr", wenn alle Einzelbedingungen, die in Klammern eingeschlossen werden müssen, erfüllt sind.

Beispiele:

(a)
```
10 LET A = 5
20 LET B = 7
30 IF (A = 5) * (B = 7) THEN 90
```

Diese mit * zusammengesetzte Bedingung ist "wahr", weil beide Einzelbedingungen erfüllt sind. Es erfolgt daher der Sprung nach Zeile 90.

(b)
```
10 LET A = 5
20 LET B = 6
30 IF (A = 5) * (B = 7) THEN 90
40 ...
```

Diese mit * zusammengesetzte Bedingung ist nicht "wahr", weil die zweite Einzelbedingung nicht erfüllt ist. Der Programmlauf setzt daher mit dem Befehl in Zeile 40 fort.

---

1) Das Adjektiv "inklusiv" bedeutet, daß eine mit diesem ODER formulierte zusammengesetzte Bedingung "wahr" ist, wenn entweder die eine oder die andere Bedingung wahr ist oder wenn beide Bedingungen wahr sind.

2) Das "logische UND" ist bei vielen anderen BASIC-Dialekten in Anlehnung an die englische Sprache mit AND und das "logische inklusive ODER" mit OR zur formulieren.

Logisches inklusives ODER: +

Die mit + zusammengesetzte Bedingung ist dann "wahr", wenn mindestens eine der Einzelbedingungen erfüllt ist.

Beispiele:

(a)

```
10 LET A = 5
20 LET B = 6
30 IF (A = 5) + (B = 7) THEN 90
```

Diese mit + zusammengesetzte Bedingung ist "wahr", weil die erste Einzelbedingung erfüllt ist. Es erfolgt daher der Sprung nach Zeile 90.

(b)

```
10 LET A = 4
20 LET B = 6
30 IF (A = 5) + (B = 7) THEN 90
40 ...
```

Diese mit + zusammengesetzte Bedingung ist nicht "wahr", weil keine der Einzelbedingungen erfüllt ist. Der Programmlauf setzt daher mit dem Befehl in Zeile 40 fort.

PE 18.3

Versuchen Sie herauszufinden, worin der Unterschied hinsichtlich der Auswirkung zwischen den folgenden beiden Mehrfachbedingungen besteht.

(1)
```
10 LET A = 5
20 LET B = 6
 .
 .
 .
100 IF (A = 5) + (B = 7) THEN 200
 .
 .
 .
```

(2)
```
10 LET A = 5
20 LET B = 6
 .
 .
 .
100 IF (A = 5) * (B = 7) THEN 200
 .
 .
 .
```

Der Auswirkungsunterschied der beiden Mehrfachbedingungen besteht darin, daß

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

LÖS 18.3

Die in **PE 18.3** aufgeführten Mehrfachbedingungen unterscheiden sich hinsichtlich ihrer Auswirkung dahingehend, daß bei der unter

Punkt (1) genannten Mehrfachbedingung (Kombination mit + / logisches inklusives ODER) der Sprung nach Zeile 200 ausgeführt wird, weil eine der beiden Einzelbedingungen erfüllt ist.

Bei der unter

Punkt (2) genannten Mehrfachbedingung (Kombination mit * / logisches UND) erfolgt der Sprung nach Zeile 200 hingegen nicht, weil die zweite Einzelbedingung nicht erfüllt ist. Der Programmlauf setzt daher mit dem Befehl in Zeile 110 fort.

Sollte Ihre Lösung nicht richtig sein, kehren Sie zurück zur → **LE 18.3** !

ÜBUNGsprogramm zu Lektion 18

Problemstellung:

Mit Hilfe des **ÜBUNG**sprogramms zu Lektion 16 wurde (beispielsweise) der folgende verschlüsselte Text erstellt, der wieder decodiert werden soll.

```
FMTCFUI-!BMMFSMJFCTUF-
EV!CSVTUMBU[!LBMUFS!IFS[FO-
EFS!MJFCF![VOEFS!EV-
HPMETDIBDIUFM!FEMFS![JFS-
EFS!TFVG[FS!CMBTFCBMH-
EFT!USBVFSOT!MPFTDIQBQJFS-
TBOECVFDITF!NFJOFS!QFJO
VOE!CBUNPFM!NFJOFS!TDINFS[FO/
JO!FXJHFS!MJFCF!EFJO!H/!P/
```

Programmbeschreibung (Decodierung):

Zur Decodierung eines Textes, das mit Hilfe des **ÜBUNG**sprogramms zu Lektion 16 verschlüsselt wurde, kann man dasselbe Programm verwenden, das nur an einer Stelle (Zeile 2080) zu ändern ist, und zwar dort nur ein einziges Zeichen ("-" statt "+").

Wenn man das Decodierungsprogramm zusätzlich zum Verschlüsselungsprogramm speichern will, ist es aus Gründen der Klarheit jedoch empfehlenswert, einige Textstellen in REM-Anweisungen und bestimmte Strings anzupassen.

Programmliste:

Die folgende Programmliste enthält nur jene Befehle, die man im **ÜBUNG**sprogramm zu Lektion 16 ändern muß.

```
10 REM **************
20 REM * DECODIERUNG *
30 REM **************
 .
 .
 .
100 GOSUB 2000 : REM TEXT-ENTSCHLUESS.
 .
 .
 .
1060 PRINT " BITTE DEN VERSCHLUESS. TEXT
 EINGEBEN?"
 .
 .
 .
1110 PRINT " DAMIT DER ENTSCHLUESSELTE T
EXT AUF DEM"
 .
 .
 .
2000 REM TEXT-ENTSCHLUESSELUNG
 .
 .
2030 PRINT " ENTSCHLUESSELTER TEXT:"
 .
 .
 .
2080 LET C = ASC(B$) - 1
```

19. Beendigung der Programmausführung mit END

Unterbrechung der Programmausführung mit STOP

Wiederaufnahme der Programmausführung mit CONT

LE 19

Die Anweisung

END (von engl. "to end" = beenden),

die wir schon einige Male verwendet haben, beendet den Programmlauf. Bei manchen BASIC-Rechnern muß sie unter der höchsten Zeilennummer des Programms aufgeführt sein, bei anderen Systemen - wie beispielsweise beim SHARP MZ-700 - kann sie an beliebiger Stelle erscheinen und am Programmende auch fortgelassen werden.

Beispiel:

```
 .
 .
950 IF Z > 100 THEN END
960 GOTO 700
```

Bei Auftreten des Befehls

STOP (von engl. "to stop" = anhalten)

wird der Programmlauf mit der folgenden Systemmeldung beendet:

```
Break in ...        (= Unterbrechung in Zeile ...)
Ready
```

Die STOP-Anweisung kann an mehreren Stellen des Programms stehen, wenn mehrere Beendigungspunkte vorgesehen sind. Die Programmausführung wird aber fortgesetzt, wenn der Benutzer das Systemkommando

CONT (von engl. "to continue = fortfahren, fortsetzen)

erteilt. Man verwendet den BASIC-Befehl S T O P und das Systemkommando C O N T vor allem, um Programmfehler zu suchen. Denn nach der Unterbrechung der Programmausführung ist es möglich, die Inhalte (u. U. Zwischenergebnisse) der Variablen zu erfragen und / oder zu verändern. Das geschieht im direkten Modus, d. h. die entsprechenden Befehle werden wie Systemkommandos (ohne Zeilennumerierung!) erteilt.

Beispiel:

```
10 REM ********************************
20 REM * BEISPIEL FUER VERWENDUNG VON *
30 REM *          STOP UND CONT        *
40 REM ********************************
50 CLS
60 INPUT "DM-BETRAG EINGEBEN ! "; KO
70 PRINT
80 REM ZINSBERECHNUNG
90 REM --------------
100 LET Z = KO * 7.5 / 100
110 STOP
120 LET KG = KO + Z
130 PRINT "ENDBETRAG (MIT ZINSEN):"; KG
140 END
RUN
```

```
DM-BETRAG EINGEBEN ! 350 <-----Benutzereingabe

Break in 110              <-----Systemmeldung
Ready                     <-----      "
PRINT Z                   <-----Befehl im direkten Modus
 26.25                    <-----Inhalt von Z
Ready                     <-----Systemmeldung
CONT                      <-----Systemkommando des Benutzers
ENDBETRAG (MIT ZINSEN): 376.25
Ready
```

PE 19

Welche der folgenden Behauptungen ist/sind richtig?

A Die END-Anweisung beendet den Programmlauf.

B Bei manchen BASIC-Rechnern muß die END-Anweisung unter der höchsten Zeilennummer des Programms aufgeführt sein, bei anderen Systemen - so beispielsweise beim SHARP MZ-700 - kann sie an beliebiger Stelle erscheinen und am Ende auch fortgelassen werden.

C Bei Auftreten einer END-Anweisung wird der Programmlauf mit der folgenden Systemmeldung beendet:

```
CONT
```

D Bei Auftreten einer STOP-Anweisung wird der Programmlauf mit der folgenden Systemmeldung beendet:

```
Break in ...
Ready
```

E Der STOP-Befehl muß unter der höchsten Zeilennummer des Programms aufgeführt sein.

F Der STOP-Befehl kann an mehreren Stellen des Programms stehen, wenn mehrere Beendigungspunkte vorgesehen sind.

G Die durch einen STOP-Befehl unterbrochene Programmausführung wird fortgesetzt, wenn der Benutzer das Systemkommando CONT erteilt.

H Der BASIC-Befehl S T O P und das Systemkommando C O N T werden vor allem verwendet, um Programmfehler zu suchen.

I Der BASIC-Befehl S T O P beseitigt in Zusammenwirkung mit dem Systemkommando C O N T automatisch formale und logische Programmfehler.

J Nach der Unterbrechung der Programmausführung durch den STOP-Befehl ist es möglich, die Inhalte (u. U. Zwischenergebnisse) der Variablen zu erfragen und/oder zu verändern.

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

LÖS 19

Die Lösungsbuchstaben lauten

A, B, D, F, G, H, J .

Sollte Ihre Lösung nicht richtig sein, kehren Sie zurück zur
→ **LE 19** !

ÜBUNGsprogramm zu Lektion 19

Programmbeschreibung (Lottozahlen):

Das folgende Programm ermittelt sechs ganze Zufallszahlen im Bereich 1 - 49 (= Lottozahlen). Auf Wunsch des Anwenders wird der Vorgang wiederholt.

Programmliste:

```
10 REM ***************
20 REM * LOTTOZAHLEN *
30 REM ***************
40 REM
50 DIM Z(6)
60 REM
70 REM INFORMATION U. EINGABEN
80 REM =======================
90 CLS
100 PRINT
110 PRINT " UM MIT HILFE MEINES ZUFALLSG
ENERATORS"
120 PRINT
130 PRINT " SPEZIELL FUER SIE 'IHRE'"
140 PRINT
150 PRINT " * LOTTO-GLUECKSZAHLEN * "
160 PRINT
170 PRINT " ERMITTELN ZU KOENNEN,"
180 PRINT
190 PRINT " BENOETIGE ICH DIE FOLGENDEN
ANGABEN:"
200 PRINT : PRINT : PRINT
210 PRINT " FAMILIENNAME"
220 PRINT
230 PRINT " GESCHLECHT (M/W)"
240 PRINT
```

```
250 PRINT " GEBURTSDATUM"
260 PRINT " (TT.MM.JJ)"
270 CURSOR 18, 13
280 INPUT NA$
290 CURSOR 18, 15
300 PRINT "? "
310 GET MW$
320 IF (MW$<>"M") * (MW$<>"W") THEN 310
330 CURSOR 20, 15
340 PRINT MW$
350 CURSOR 18, 17
360 INPUT GB$
370 REM UEBERPRUEFUNG DES EINGEGEBENEN
380 REM GEBURTSDATUMS:
390 REM ------------------------------
400 IF LEN(GB$) <> 8 THEN 630
410 IF MID$(GB$, 3, 1) <> "." THEN 630
420 IF MID$(GB$, 6, 1) <> "." THEN 630
430 LET TT$ = LEFT$(GB$, 2)
440 LET TT  = VAL(TT$)
450 IF (TT < 1) + (TT > 31) THEN 630
460 LET MM$ = MID$(GB$, 4, 2)
470 LET MM  = VAL(MM$)
480 IF (MM < 1) +  (MM > 12) THEN 630
490 REM
500 REM EINE UEBERPRUEFUNG DES GEBURTS-
510 REM JAHRES ERSCHEINT NICHT SINN-
520 REM VOLL, DA ALLE 2-STELLIGEN
530 REM ZAHLEN RICHTIG SEIN KOENNEN.
540 REM DER NUMERISCHE WERT DES GE-
550 REM BURTSJAHRES WIRD DAHER NUR ER-
560 REM MITTELT, UM DIE 'INDIVIDUELLE
570 REM GLUECKSZAHL' BERECHNEN ZU
580 REM KOENNEN.
590 REM -------------------------------
600 LET JJ$ = RIGHT$(GB$, 2)
610 LET JJ  = VAL(JJ$)
620 GOTO 710
630 REM LOESCHUNG DES GEBURTSDATUMS U.
640 REM FEHLERMELDUNG
650 CURSOR 18, 17
660 PRINT "EINGABEFEHLER !!!!!!!"
670 FOR I = 1 TO 500 : NEXT I
```

```
680 CURSOR 18, 17
690 PRINT "                           "
700 GOTO 350
710 PRINT : PRINT : PRINT
720 REM BESTAETIGUNG DER EINGABEN:
730 REM -------------------------
740 PRINT TAB(15) "EINGABEN RICHTIG ? (J
/N)"
750 GET A$
760 IF A$ = "J" THEN 800
770 IF A$ = "N" THEN 70
780 GOTO 750
790 REM
800 REM ERMITTLUNG DER INDIVIDUELLEN
810 REM GLUECKSZAHL:
820 REM ============================
830 REM
840 LET GZ = TT + MM + JJ
850 REM
860 REM BEKANNTGABE DER INDIVIDUELLEN
870 REM GLUECKSZAHL:
880 REM =============================
890 CLS
900 PRINT : PRINT : PRINT
910 PRINT " IHRE 'INDIVIDUELLE GLUECKSZA
HL', MIT"
920 PRINT
930 PRINT " DER DER ZUFALLSGENERATOR BEE
INFFLUSST"
940 PRINT
950 PRINT " WERDEN SOLL, LAUTET:
960 FOR I = 1 TO 5 : PRINT : NEXT I
970 PRINT TAB(12) "*** "; GZ; "  ***"
980 FOR I = 1 TO 8 : PRINT : NEXT I
990 PRINT TAB(20) "'W'-TASTE DRUECKEN!"
1000 GET A$ : IF A$ <> "W" THEN 1000
1010 REM
1020 REM 'BEEINFLUSSUNG' DES ZUFALLS-
1030 REM GENERATORS MIT DER GLUECKSZAHL
1040 REM ============================
1050 CLS
1060 FOR I = 1 TO 6 : PRINT : NEXT I
1070 PRINT TAB(10) "ACHTUNG !!!"
1080 PRINT
```

```
1090 PRINT " DER ZUFALLSGENERATOR WIRD M
IT DER"
1100 PRINT
1110 PRINT " GLUECKSZAHL 'POSITIV BEEINF
LUSST' !!"
1120 FOR I = 1 TO GZ
1130 CURSOR 12, 16
1140 PRINT I
1150 LET ZZ = RND(1)
1160 FOR K = 1 TO 100 : NEXT K
1170 NEXT I
1180 REM
1190 REM BERECHNUNG UND AUSGABE DER
1200 REM LOTTOZAHLEN:
1210 REM ==========================
1220 CLS
1230 PRINT : PRINT : PRINT
1240 IF MW$ = "W" THEN LET AN$ = " SEHR
GEEHRTE/S FRAU/FRAEULEIN " : GOTO 1260
1250 LET AN$ = " SEHR GEEHRTER HERR "
1260 PRINT AN$; NA$; " !"
1270 PRINT : PRINT
1280 PRINT " ICH EMPFEHLE FOLGENDE LOTTO
ZAHLEN: "
1290 PRINT : PRINT : PRINT
1300 REM
1310 REM 1. LOTTOZAHL:
1320 REM -------------
1330 LET I = 1
1340 LET Z(1) = INT(RND(1) * 49 + 1)
1350 PRINT "  "; Z(I);
1360 REM
1370 REM 2. - 6. LOTTOZAHL:
1380 REM -----------------
1390 FOR I = 2 TO 6
1400 LET P = 1
1410 LET Z(I) = INT(RND(1) * 49 + 1)
1420 IF P < I THEN 1450
1430 GOTO 1590
1440 REM
1450 REM PRUEFUNG, OB DIE ZAHL SCHON
1460 REM GEZOGEN WURDE:
1470 REM --------------------------
1480 IF Z(P) = Z(I) THEN 1550
1490 REM
```

```
1500 REM NEIN-ZWEIG:
1510 REM -----------
1520 LET P = P + 1
1530 GOTO 1420
1540 REM
1550 REM JA-ZWEIG:
1560 REM ---------
1570 GOTO 1400
1580 REM
1590 REM AUSGABE DER 2. - 6. LOTTOZAHL:
1600 REM -----------------------------
1610 PRINT "    "; Z(I);
1620 NEXT I
1630 FOR I = 1 TO 8 : PRINT : NEXT I
1640 PRINT TAB(2) "'W'-TASTE DRUECKEN!"
1650 GET A$
1660 IF A$ <> "W" THEN 1650
1670 REM
1680 REM AUSGABE AUF PLOTTER-DRUCKER ?
1690 REM ============================
1700 CLS
1710 PRINT : PRINT : PRINT
1720 PRINT " AUSDRUCK ERWUENSCHT ?"
1730 PRINT " (J/N)"
1740 GET A$
1750 IF A$ = "N" THEN 2050
1760 IF A$ = "J" THEN 1780
1770 GOTO 1740
1780 REM AUSGABE AUF PLOTTER-DRUCKER
1790 REM --------------------------
1800 PCOLOR 0 : MODE TN
1810 PRINT/P AN$; NA$; " !"
1820 PRINT/P
1830 PRINT/P " ICH, SHARP MZ-700, EMPFEH
LE IHNEN"
1840 PRINT/P " UNTER BERUECKSICHTIGUNG I
HRER"
1850 PRINT/P " GLUECKSZAHL"
1860 PCOLOR 1
1870 PRINT/P TAB(17) GZ; " ,"
1880 PCOLOR 0
1890 PRINT/P " DIE UNTER HERANZIEHUNG IH
RES GEBURTS-"
1900 PRINT/P " DATUMS VOM"
```

```
1910 PCOLOR 2
1920 PRINT/P TAB(15) GB$
1930 PCOLOR 0
1940 PRINT/P " ERMITTELT WURDE, DIE"
1950 PRINT/P " FOLGENDEN LOTTOZAHLEN:"
1960 PRINT/P
1970 PCOLOR 3
1980 FOR I = 1 TO 6
1990 PRINT/P  Z(I); "    ";
2000 NEXT I
2010 PRINT/P : PRINT/P
2020 PCOLOR 0
2030 PRINT/P " MIT DEN BESTEN WUENSCHEN"
2040 PRINT/P " IHR G. O. HAMANN"
2050 REM WEITERE ZAHLEN ?
2060 REM ================
2070 CLS
2080 FOR I = 1 TO 10 : PRINT : NEXT I
2090 PRINT " WEITERE ZAHLEN ERWUENSCHT ?
 (J/N)"
2100 GET A$
2110 IF A$ = "J" THEN 70
2120 IF A$ = "N" THEN 2140
2130 GOTO 2100
2140 END
```

Hinweis:

Wenn die Lottozahlen in aufsteigender Sortierfolge vorliegen sollen, dann muß das Programm um eine Sortierroutine (ähnlich dem **ÜBUNG**sprogramm zu Lektion 20) erweitert werden, die nach der Ermittlung der Lottozahlen aber vor deren Ausgabe zu durchlaufen ist.

20. Schleifenbildung mit FOR .. TO .. (STEP ..) / NEXT ..

LE 20.1

Innerhalb eines Programms ist es häufig erforderlich, daß eine Reihe von Befehlen mehrmals nacheinander durchlaufen wird. Man bezeichnet sie als Programmschleife. Eine solche Programmschleife enthielt beispielsweise unser Programmbeispiel auf Seite 8-07:

```
.
.
.
90 LET A = 10
100 LET A = A + 1          }
110 LET B = A * A          }
120 PRINT A$; A; B$; B     }  Programmschleife
130 IF A < 100 THEN 100    }
.
.
.
```

Der Befehl in Zeile 90 (LET A = 10) erfüllt eine vorbereitende Aufgabe für die eigentliche Schleife, die in Zeile 100 beginnt. Die Variable A wird dann innerhalb der Schleife immer wieder um 1 erhöht, das Quadrat von A gebildet und das (erläuterte) Ergebnis ausgegeben. Dieser Zyklus wird so oft wiederholt, bis die Bedingung der Zeile 130 (A < 100) nicht mehr erfüllt ist und die Schleifendurchläufe beendet sind.

Anders ausgedrückt (nämlich in Anlehnung an einen neuen BASIC-Befehl, mit dem man überlicherweise eine Schleife codiert):

Die Schleifenbefehle werden f ü r alle (Ganzzahl-) Werte ausgeführt, die A von 11 bis 100 (also für 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, . . . 98, 99, 100) annehmen kann.

Codierung der Schleife mit
F O R .. T O .. (S T E P ..) / N E X T .. (von engl. "for" = für; von engl. "to" = bis; von engl. "step" = Schritt, Schrittweite; von engl. "next" = nächstes)

```
.
.
.
90 FOR A = 11 TO 100 STEP 1      ◀ Beginn der Schleife
100 LET B = A * A
110 PRINT A$; A; B$; B
120 NEXT A                       ◀ Ende der Schleife
.
.
.
```

E r l ä u t e r u n g e n :

(1) Zwischen den Anweisungen F O R .. (Zeile 90) und N E X T .. (Zeile 120) sind die Befehle anzuordnen, die der Rechner mehrmals (= in der Schleife) bearbeiten soll.

(2) Die Befehle zwischen der F O R - und N E X T - Anweisung werden zunächst mit dem Anfangswert der Laufvariablen (= 11) ausgeführt. Dann werden die Schrittweite (= step) entsprechend der Angabe (= 1) zur Laufvariablen addiert und die Ausführung der Befehle zwischen F O R .. und N E X T .. wiederholt. Dies geschieht so lange, bis die Laufvariable den Endwert (= 100) überschritten hat. Der Programmlauf wird dann mit jener Anweisung fortgesetzt, die auf die N E X T - Anweisung folgt.

(3) Die Laufvariable in der F O R - Anweisung muß numerisch sein und mit jener in der N E X T - Anweisung übereinstimmen.

(4) Wenn der Programmierer die Schrittweite (= step) nicht angibt, dann wird diese mit +1 unterstellt. (Demnach ist die Angabe S T E P 1 - wie in Zeile 90 des Beispiels auf Seite 20-02 - in jedem Fall entbehrlich.)

(5) Sowohl für den Laufbereichsanfangswert (Beispiel: .. FOR X = 1 ...) und den Laufbereichsendwert (Beispiel: .. FOR X = 1 TO 20 ...) als auch für die Schrittweite (Beispiel: .. FOR X = 1 TO 20 STEP 2) ist ebenfalls die Angabe von Variablen und mathematischen Ausdrücken zulässig (Beispiel: .. FOR I = A TO B/2 STEP X).

(6) Die Laufvariable kann innerhalb der Schleife für verschiedene Operationen herangezogen werden, nur sollte man ihren Inhalt nicht verändern.

(7) Wenn man eine negative Schrittweite angibt (Beispiel: ... STEP -1), dann muß der Endwert der Laufvariablen naturgemäß niedriger als deren Anfangswert sein (Beispiel: .. FOR I = 20 TO 10 STEP -1).

(8) Wir empfehlen, nur mit ganzzahligen Laufbereichsgrenzen und Schrittweiten zu arbeiten.

(9) Beim SHARP MZ-700 braucht die Laufvariable in der NEXT-Anweisung nicht unbedingt angegeben zu werden. Aus Gründen der Übersichtlichkeit empfehlen wir, hiervon keinen Gebrauch zu machen.

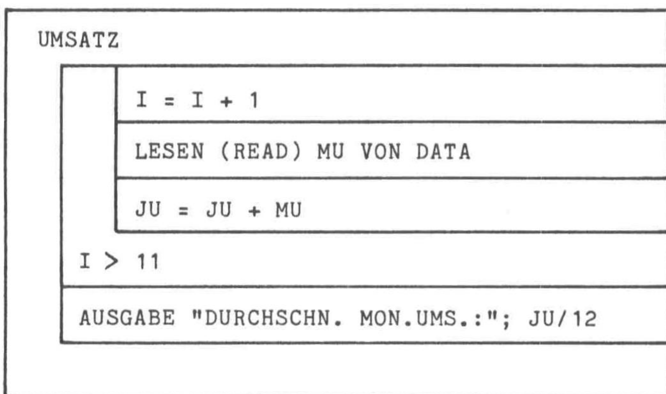
PE 20.1

In **LE 14** hatten wir die folgende Aufgabe behandelt (vgl. Seite 14-01 f. !):

(1) Aufgabendefinition (1. Teil)

Aus den 12 Monatsumsätzen einer Unternehmung ist der durchschnittliche Monatsumsatz zu berechnen. (Die Monatsumsätze sind nacheinander mit dem READ-Befehl der Variablen MU zuzuweisen und mit Hilfe der Variablen JU zu addieren. Die Division von JU durch 12 ergibt den durchschnittlichen Monatsumsatz. Das Ergebnis ist auszugeben.)

(2) Reihenfolgeplanung (Struktogramm) / 1. Teil



Ergänzen Sie in dem folgenden Programm in Anlehnung an das nebenstehende Struktogramm (vgl. Seite 20-04) die fehlenden Befehle, wobei - abweichend von der Lösung auf Seite 14-03 - für die Schleife der F O R - und der N E X T - Befehl zu verwenden sind. (Dadurch wird eine Befehlszeile weniger benötigt!)

Hinweis: Überlegen Sie zuerst, wie häufig die eigentlichen Schleifenbefehle auszuführen sind.

```
10 REM * UMSATZ *

20 ..............................

30 ..............................

40 ..............................

50 ..............................

60 PRINT "DURCHSCHN. MON.UMS.:"; JU/12
70 DATA 99485, 76356, 78999
80 DATA 45384, 66666, 77333
90 DATA 48956, 55432, 86123
100 DATA 98765, 56734, 34575
110 END
```

LÖS 20.1

Programm zum Struktogramm der **PE 20.1** (Seite 20-04) unter Verwendung des F O R - und des N E X T - Befehls:

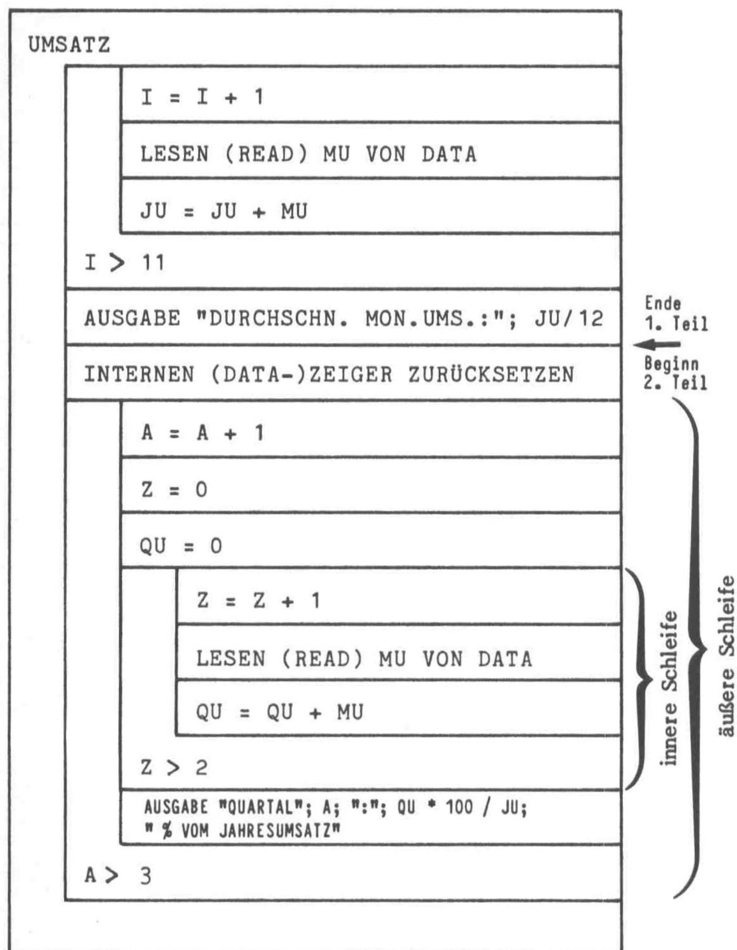
```
10 REM * UMSATZ *
20 FOR I = 1 TO 12
30 READ MU
40 LET JU = JU + MU
50 NEXT I
60 PRINT "DURCHSCHN. MON.UMS.:"; JU/12
70 DATA 99485, 76356, 78999
80 DATA 45384, 66666, 77333
90 DATA 48956, 55432, 86123
100 DATA 98765, 56734, 34575
110 END
```

Sollte Ihre Lösung nicht richtig sein, kehren Sie zurück zur → **LE 20.1** !

LE 20.2

Im 2. Teil der Aufgabe in **LE 14** (vgl. Seite 14-01 und 14-04 f.) sollte ermittelt werden, wieviel Prozent die Quartalsumsätze vom Gesamtumsatz ausmachen. Deshalb setzen wir zunächst den internen "DATA-Zeiger" mit dem RESTORE-Befehl zurück und bildeten dann zwei Schleifen - eine innere und eine äußere:

Reihenfolgeplanung (Struktogramm) 1. und 2. Teil



Zuzuweisende Daten:

99485, 76356, 78999, 45384, 66666, 77333
48956, 55432, 86123, 98765, 56734, 34575

Auch für die Codierung des nebenstehenden Struktogramms ist es sinnvoll - abweichend von der Lösung auf Seite 14-05 - FOR-NEXT-Schleifen zu bilden, denn in BASIC ist das Ineinanderschachteln von Schleifen zulässig. Anders ausgedrückt: In eine FOR-NEXT-Schleife kann eine andere eingebettet sein.

Codierung des Struktogramms mit zwei FOR-NEXT-Schleifen

```
10 REM * UMSATZ *
20 FOR I = 1 TO 12
30 READ MU
40 LET JU = JU + MU
50 NEXT I
60 PRINT "DURCHSCHN. MON.UMS.:"; JU/12
70 RESTORE
80 FOR A = 1 TO 4
90 LET QU = 0
100 FOR Z = 1 TO 3
110 READ MU
120 LET QU = QU + MU
130 NEXT Z
140 PRINT "QUARTAL"; A; ":"; QU * 100 /
JU; " % VOM JAHRESUMSATZ"
150 NEXT A
160 DATA 99485, 76356, 78999
170 DATA 45384, 66666, 77333
180 DATA 48956, 55432, 86123
190 DATA 98765, 56734, 34575
200 END
```

Erläuterungen:

(1) Der Befehl ".. LET Z = 0" ist hier entbehrlich, da durch die FOR-Anweisung sichergestellt ist, daß die Variable Z immer den festgelegten Anfangswert erhält, wenn die innere Schleife erneut aufgerufen wird.

(2) Anders verhält es sich mit der Variablen QU. Sie muß vor Beginn der inneren Schleife auf 0 gesetzt werden, weil wir mit diesem Datenfeld viermal die jeweiligen Quartalsumsätze addieren wollen.

Es muß hervorgehoben werden, daß beim Ineinanderschachteln von Schleifen

- die Namen der Schleifenvariablen sich unterscheiden müssen und

- die NEXT-Anweisung der inneren Schleife vor der NEXT-Anweisung der äußeren Schleife aufzuführen ist.

Beispiele:

(1) Zulässige Konstruktionen

```
┌──FOR A =
│  ┌──FOR B =
│  │  ┌──FOR C =
│  │  └──NEXT C
│  └──NEXT B
└──NEXT A
```

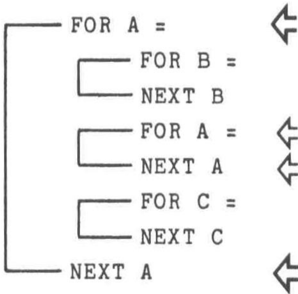
```
┌──FOR A =      ┐
│  ┌──FOR B =   │
│  └──NEXT B    │
│  ┌──FOR C =   │
│  └──NEXT C    │
└──NEXT A       ┘
```

Hierfür ist beim SHARP MZ-700 auch die folgende Schreibweise zulässig:

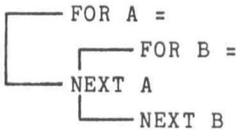
```
┌──FOR A =
│  ┌──FOR B =
│  └──NEXT B
│  ┌──FOR C =
└──┴──NEXT C, A
```

↑ Reihenfolge der Variablen unbedingt beachten !

(2) Unzulässige Konstruktionen



Diese Konstruktion ist unzulässig, weil sich die Laufvariablen der äußeren und einer inneren Schleife nicht unterscheiden.



Diese Konstruktion ist unzulässig, weil die NEXT-Anweisung der inneren Schleife (B) nach der NEXT-Anweisung der äußeren Schleife (A) erscheint. (Die Schleifen "schneiden" sich.)

PE 20.2

Erläutern Sie die Fehler in den folgenden FOR-NEXT-Schleifen!

(1)
```
10 FOR A = F TO F STEP 2
 .
 .
50 NEXT A
```

Fehler: . . . .

. . . .

(2)
```
10 FOR A = 1 TO 20 STEP -1
 .
 .
60 NEXT A
```

Fehler: . . . .

. . . .

(3)
```
10 FOR A = 1 TO 50
 .
 .
50 FOR B = 1 TO 10
 .
 .
80 NEXT A
 .
 .
160 NEXT B
```

Fehler: ....

....

(4)
```
10 FOR F1 = A TO B STEP C/2
 .
 .
50 FOR F2 = 1 TO 10
 .
 .
80 NEXT F2
 .
 .
150 FOR F1 = 1 TO 70
 .
 .
190 NEXT F1
 .
 .
280 NEXT F1
```

Fehler: ....

....

LÖS 20.2

Die FOR-NEXT-Schleifen der **PE 20.2** enthielten die folgenden Fehler:

(1) Die Schleife wird nur einmal durchlaufen; sie ist daher überflüssig.

(2) Bei negativer Schrittweite muß der Endwert der Laufvariablen naturgemäß niedriger angegeben werden als deren Anfangswert.

(3) Diese Konstruktion ist unzulässig, weil die NEXT-Anweisung der inneren Schleife (B) nach der NEXT-Anweisung der äußeren Schleife (A) erscheint. Die Schleifen "schneiden" sich.

(4) Die Laufvariable der zweiten inneren Schleife (F1) unterscheidet sich nicht von der Laufvariablen der äußeren Schleife (F1).

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 20.1** !

ÜBUNGsprogramm zu Lektion 20

Programmbeschreibung (Sortierroutine):

Das folgende Programm dient der alphabetischen Sortierung beliebiger Strings.

Zunächst ist eine von Anwendungsfall zu Anwendungsfall differierende Anzahl von Begriffen in ein Feld (hier: T$(..) einzulesen, dessen Größe zu Beginn des Programms festgelegt wird.

Programmliste (Teil 1):

```
10 REM ******************
20 REM * SORTIERROUTINE *
30 REM ******************
40 CLS
50 PRINT
60 PRINT " WIEVIEL BEGRIFFE WOLLEN SIE E
INGEBEN?"
70 PRINT
80 PRINT " BITTE GGF. SCHAETZEN!"
90 PRINT : PRINT
100 PRINT " ACHTUNG: SCHAETZUNG DARF NIC
HT ZU"
110 PRINT
120 PRINT " NIEDRIG SEIN, DA SONST NICHT
 GENUEGEND"
130 PRINT
140 PRINT " ELEMENTE RESERVIERT WERDEN."
150 PRINT : PRINT
160 PRINT " WIEVIEL BEGRIFFE ";
170 INPUT A
```

Da das erste Element des Feldes - also T$(0) - zunächst unbelegt bleiben soll, ergibt sich die Anzahl der zu reservierenden Elemente aus dem Inhalt der Variablen A.

Programmliste (Teil 2):

```
180 DIM T$(A)
```

Nach der Reservierung des Feldes T$(..) werden in einer FOR .. NEXT-Schleife die unsortierten Strings eingegeben.

Programmliste (Teil 3):

```
190 CLS
200 PRINT
210 PRINT " PRO ZEILE NUR EINEN BEGRIFF
EINGEBEN!"
220 PRINT : PRINT : PRINT
230 FOR I = 1 TO A
240 PRINT " BEGRIFF"; I; " ";
250 INPUT T$(I)
260 PRINT
270 NEXT I
```

Mit den folgenden Befehlen werden die eingegebenen Begriffe in alphabetischer Reihenfolge geordnet. Zu diesem Zweck vergleichen wir das Element T$(1) mit den restlichen auf "kleiner/gleich".

Ein solcher Vergleich ist möglich, weil alle Zeichen - also auch die Buchstaben - im Arbeitsspeicher als eine Folge von (Binär-)Ziffern dargestellt sind, wobei die wertmäßige Verschlüsselung so vorgenommen wurde, daß sie der Reihenfolge des Alphabets entspricht (vgl. hierzu auch Anhang-25 f.); demnach hat "A" den niedrigsten Wert und "Z" den höchsten.

Sobald die Bedingung nicht erfüllt ist, werden die Inhalte der Feldelemente getauscht, wobei als Zwischenspeicher das Element T$(0) dient.

Beispiel:

	BUEROKLAMMERN	KLEBESTIFTE	BLEISTIFTE	FARBBAENDER	SCHREIBWERKE
T$(0)	T$(1)	T$(2)	T$(3)	T$(4)	T$(5)

1. Durchlauf:

1. Vergleich: T$(1) mit T$(2): Bedingung erfüllt
2. Vergleich: T$(1) mit T$(3): Bedingung nicht erfüllt, daher Feldertausch:
   T$(0) = T$(1)
   T$(1) = T$(3)
   T$(3) = T$(0)
3. Vergleich: T$(1) mit T$(4): Bedingung erfüllt
4. Vergleich: T$(1) mit T$(5): Bedingung erfüllt

Nach Abschluß des 1. Durchlaufs ist im Element T$(1) der Begriff "BLEISTIFTE" gespeichert. Allgemein ausgedrückt: Nach Abschluß des 1. Durchlaufs ist im Element T$(1) der niedrigstwertige Begriff der zu sortierenden Bezeichnungen.

2. Durchlauf:

Jetzt wird T$(2) mit T$(3),
T$(2) mit T$(4),
T$(2) mit T$(5)
verglichen und ggf. die Felder getauscht, so daß nach dem 2. Durchlauf in T$(2) der niedrigstwertige Begriff der Elemente T$(2) - T$(5) ist.

3. Durchlauf:

Nun wird T$(3) mit T$(4),
T$(3) mit T$(5)
verglichen. Ggf. erfolgt ein Elementetausch analog wie oben.

4. Durchlauf:

Die Programmroutine hat ihre Aufgabe nach dem 4. Durchlauf erfüllt, wenn nämlich

T$(4) mit T$(5)

verglichen worden ist und ggf. ein Tausch der Elemente erfolgte.

Programmliste (Teil 4):

```
280 FOR I = 1 TO (A - 1)
290 FOR Y = (I + 1) TO A
300 IF T$(I) <= T$(Y) THEN 340
310 LET T$(0) = T$(I)
320 LET T$(I) = T$(Y)
330 LET T$(Y) = T$(0)
340 NEXT Y
350 NEXT I
```

Um überprüfen zu können, ob das Programm wunschgemäß arbeitet, sollen abschließend die Elemente des Feldes T$(..) in einer FOR .. NEXT-Schleife ausgegeben werden.

Programmliste (Teil 5):

```
360 CLS
370 PRINT
380 PRINT " SORTIERTE BEGRIFFE:"
390 PRINT : PRINT
400 FOR I = 1 TO A
410 PRINT TAB(10) T$(I)
420 PRINT
430 NEXT I
440 END
```

21. Unterprogramm-Technik mit GOSUB / RETURN

LE 21

Die bisher in dieser BASIC-Einführung erläuterten Programmbeispiele waren sehr einfach und kurz. In der üblichen Programmierpraxis hingegen sind die zu lösenden Aufgaben und damit die Programme wesentlich umfangreicher und komplexer.

Um die Übersichtlichkeit bei großen Vorhaben zu gewährleisten, ist es möglich, mehrere sinnvoll zusammengehörige Operationen zu einem Unterprogramm zusammenzufassen. Die Details dieses Unterprogramms (dieser Unterprogramme) sind im Anschluß an das Hauptprogramm aufzuführen, das mit dem END-Befehl abzuschließen ist.

Die Unterprogramm-Technik sollte außerdem angewandt werden, wenn bestimmte Anweisungsfolgen mehr als einmal auftreten.

Ein Unterprogramm wird in BASIC mit dem Befehl

GOSUB (von engl. "go to subprogram" = gehe zum Unterprogramm)

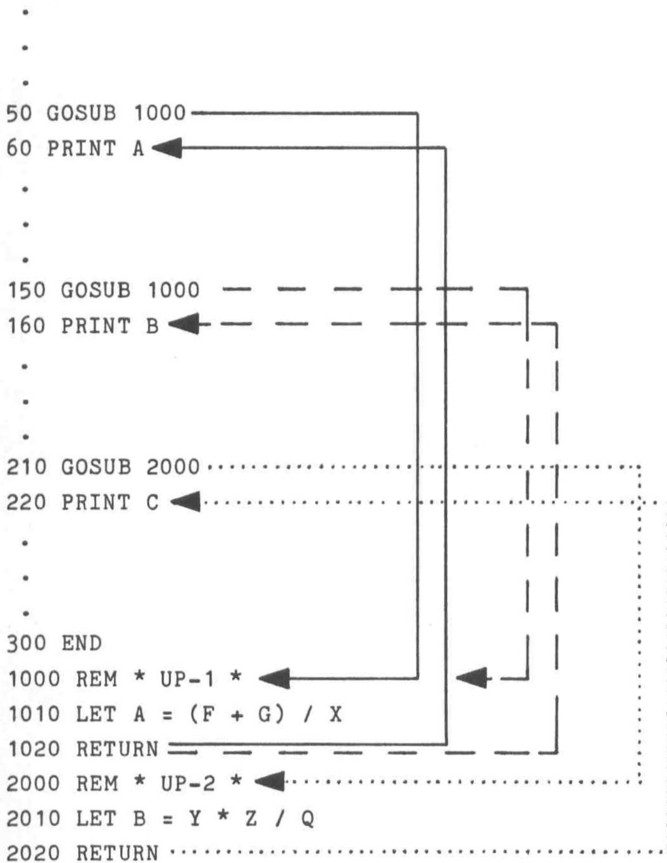
aufgerufen. Dem GOSUB-Befehl folgt unmittelbar die Zeilennummer, in der das Unterprogramm beginnt.

Am Ende des Unterprogramms muß die Anweisung

RETURN (von engl. "to return" = zurückkehren)

aufgeführt werden. Sie bewirkt, daß der Rechner zu jenem Befehl zurückspringt, der dem zuletzt bearbeiteten GOSUB-Befehl folgt.

Grafische Darstellung:



Ergänzende Hinweise:

(1) Es ist zu empfehlen, zunächst die Befehle des Hauptprogramms zu schreiben.

(2) Um zu verhindern, daß ein automatischer Überlauf in das (die) nachfolgende(n) Unterprogramm(e) erfolgt, sollte das Hauptprogramm mit dem END-Befehl abgeschlossen werden.

(3) Das Verschachteln von Unterprogrammen (= in einem Unterprogramm wird ein anderes Unterprogramm aufgerufen) ist zulässig.

(4) Ein Unterprogramm kann mehrere RETURN-Anweisungen (= mehrere "Ausgänge") aufweisen. Die zuerst erreichte RETURN-Anweisung wird ausgeführt. (Anmerkung: Es ist zu empfehlen, in einem Unterprogramm nur eine RETURN-Anweisung vorzusehen, auch wenn BASIC dies nicht verlangt. Ein Unterprogramm, das mehrere Ausgänge enthält, ist unübersichtlich und verstößt gegen die Regeln der Strukturierten Programmierung. (Vgl. hierzu die Ausführungen auf Seite 4-49!)

(5) Alle Variablen und vereinbarten Funktionen gelten gemeinsam für das Hauptprogramm und alle aufgerufenen Unterprogramme.

(6) Neben dem Aufruf eines BASIC-Unterprogramms durch GOSUB besteht zusätzlich die Möglichkeit, mit USR(Adresse) bzw. USR(Adresse, X$) eine Routine in Maschinensprache aufzurufen, die sich in einem Speicherbereich befindet, der für BASIC-Befehle durch die Anweisung LIMIT Adr. (= Adresse) gesperrt wurde. Dem Anfänger empfehlen wir, hiervon keinen Gebrauch zu machen; der fortgeschrittene Leser sei auf die Systemunterlagen des Herstellers verwiesen.

PE 21

(1) In einer Programmzeile 120 wollen Sie ein Unterprogramm aufrufen, das in Zeile 2000 beginnt. Wie lautet der entsprechende Befehl?

. . . .

(2) Mit welchem (welchen) Befehl(en) beendet man ein Unterprogramm, um ins Hauptprogramm zurückzukehren?

A STOP
B END
C RETURN
D GOSUB
E GOTO

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

(3) Um zu verhindern, daß ein automatischer Überlauf in das (die) nachfolgende(n) Unterprogramm(e) erfolgt, sollte das Hauptprogramm mit folgendem Befehl abgeschlossen werden:

. . . .

(4) Versuchen Sie, den entscheidenden Unterschied zwischen dem GOTO-Befehl und dem GOSUB-Befehl zu erläutern!

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

LÖS 21

(1) In einer Programmzeile 120 wollen Sie ein Unterprogramm aufrufen, das in Zeile 2000 beginnt. Der entsprechende Befehl lautet

120 GOSUB 2000 .

(2) Ein Unterprogramm beendet man mit dem Befehl

.. RETURN .

Der Lösungsbuchstabe lautet daher

C .

(3) Um zu verhindern, daß ein automatischer Überlauf in das (die) nachfolgende(n) Unterprogramm(e) erfolgt, sollte das Hauptprogramm mit folgendem Befehl abgeschlossen werden:

.. END .

(4) Im Gegensatz zum GOTO-Befehl merkt sich das System beim GOSUB-Befehl die "Absprungstelle", so daß der Rechner nach Ausführung des Unterprogramms zu dem Befehl zurückkehren kann, der dem jeweiligen GOSUB-Befehl folgt. So ist es beispielsweise möglich, dasselbe Unterprogramm von verschiedenen Stellen des Hauptprogramms aufzurufen und (automatisch) an diese verschiedenen Stellen zurückzuspringen.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 21** !

ÜBUNGsprogramm zu Lektion 21

Problemstellung:

Bisweilen ist zwischen zwei Personen keine Entscheidung darüber herbeizuführen, wem von beiden die günstigere Alternative zusteht. In einer solchen Situation ist es nicht unüblich, daß "Ähre" und "Zahl" eines Geldstücks je einem Beteiligten zugeordnet und aufgrund des Ergebnisses eines Münzwurfs geurteilt wird.

Wem wegen der Gewichtigkeit eines Problems die obige Lösung zu einfach ist, kann ein Programm erstellen und den Münzwurf simulieren lassen.

Programmbeschreibung ("Ähre" oder "Zahl"):

Das folgende Programm simuliert einen Münzwurf und gibt nach Betätigung der 'M'-Taste zufallsabhängig "Ähre" oder "Zahl" auf dem Bildschirm aus.

Programmliste:

```
10 REM ********************
20 REM * AEHRE ODER ZAHL *
30 REM ********************
40 CLS
50 FOR I = 1 TO 10 : PRINT : NEXT I
60 PRINT " BITTE 'M'-TASTE DRUECKEN!"
70 PRINT
80 PRINT " (DANN WIRD DER MUENZWURF SIMU
LIERT.)
90 GET A$
100 LET X = RND(X)
110 IF A$ <> "M" THEN 90
```

```
120 LET M = INT(RND(1) * 2)
130 CLS
140 FOR I = 1 TO 10 : PRINT : NEXT I
150 IF M = 0 THEN PRINT TAB(15) "*** AEH
RE ***" : GOTO 170
160 PRINT TAB(15) "*** ZAHL ***"
170 FOR I = 1 TO 10 : PRINT : NEXT I
180 PRINT TAB(15) "NOCH EINMAL ? (J/N)"
190 GET A$
200 IF A$ = "J" THEN 40
210 IF A$ = "N" THEN 230
220 GOTO 190
230 END
```

22. ON .. GOTO ..
ON .. GOSUB ..
ON ERROR GOTO .. / RESUME .. / ERN / ERL / ERROR ..

LE 22.1

Ursprünglich wurde die Programmiersprache BASIC hinsichtlich des Sprachumfangs bewußt einfach gehalten, um insbesondere Anfängern einen leichten Einstieg in die Datenverarbeitung zu ermöglichen. Inzwischen hat man jedoch - ähnlich wie bei anderen Programmiersprachen - zahlreiche Erweiterungen geschaffen. Zu diesen Erweiterungen kann man auch die ON ..-Befehle rechnen.

Die Anweisung

ON .. GOTO .. (von engl. "on", hier frei zu übersetzen mit "abhängig von", und von engl. "to go to" = gehen nach)

stellt eine Zusammenfassung von mehreren unmittelbar aufeinanderfolgenden IF .. THEN ..-Konstruktionen dar.

Beispiel für eine Mehrfachverzweigung ohne "ON .. GOTO ..":

```
 .
 .
70 IF A = 1 THEN 1000
80 IF A = 2 THEN 2000
90 IF A = 3 THEN 3000
 .
 .
```

Die drei IF .. THEN-Anweisungen können wir in einem einzigen ON .. GOTO-Befehl zusammenfassen:

Beispiel für eine Mehrfachverzweigung mit "ON GOTO ..":

```
.
.
70 ON A GOTO 1000, 2000, 3000
.
.
```

Der Befehl könnte etwa folgendermaßen übersetzt werden:

Abhängig vom Inhalt der Variablen A gehe nach Zeile 1000 oder 2000 oder 3000 (oder führe die nächste Anweisung aus)!

Ist A nämlich = 1, dann verzweigt der Rechner zur ersten angegebenen Adresse; ist A = 2, dann verzweigt der Rechner zur zweiten angegebenen Adresse; u. s. w.

Ergänzende Hinweise:

(1) Nach ON kann statt einer numerischen Variablen auch ein mathematischer Ausdruck (z. B. A * Z / 2) stehen.

(2) Durch den Wert des Ausdrucks bzw. der Variablen nach ON ist festgelegt, zu welcher der angegebenen Zeilennummern verzweigt wird.

(3) Wenn das Ergebnis des Ausdrucks oder der Inhalt der angegebenen Variablen kein ganzzahliger Wert ist, dann erfolgt immer eine Abrundung zur Ganzzahl (z. B. 4.2 und 4.9 ergeben 4).

(4) Wenn der Wert des Ausdrucks / der Variablen nach ON größer ist als die Anzahl der angegebenen Sprungadressen, dann kommt der Befehl zur Ausführung, der auf die ON .. GOTO-Anweisung folgt.

(5) Wenn der Wert des Ausdrucks / der Variablen nach ON <= 0 ist, dann gelangt wie bei Pkt. (4) der Befehl zur Ausführung, der auf die ON .. GOTO-Anweisung folgt. (Achtung: Insbesondere hier bestehen zwischen den verschiedenen BASIC-Dialekten erhebliche Unterschiede!)

PE 22.1

(1) In einem Programm stehen die folgenden Befehle:

```
.
.
430 ON HA GOTO 4000, 5000, 6000, 7000
440 PRINT "            UMSATZLISTE"
.
.
```

Was geschieht, wenn die Variable HA die folgenden Werte annimmt?

Wert der Variablen HA	Auswirkung
0	
0.3	
0.5	
0.9	
1	
1.6	
2	
3.8	
4	
5	
6000	

(2) In einem Programm stehen die folgenden Befehle:

```
.
.
500 ON GH GOTO 6000, 4000, 3000
510 PRINT "          STATISTIK"
.
.
```

Was geschieht, wenn die Variable GH die folgenden Werte annimmt?

Wert der Variablen GH	Auswirkung
6000	
4000	
3000	
0	
-1000	
1	
2	
3	
4	

LÖS 22.1

(1) In einem Programm stehen die folgenden Befehle:

```
 .
 .
430 ON HA GOTO 4000, 5000, 6000, 7000
440 PRINT "          UMSATZLISTE"
 .
 .
```

Wert der Variablen HA	Auswirkung
0	Forts. des Programmlaufs in Zeile 440
0.3	Forts. des Programmlaufs in Zeile 440
0.5	Forts. des Programmlaufs in Zeile 440
0.9	Forts. des Programmlaufs in Zeile 440
1	Sprung nach Zeile 4000
1.6	Sprung nach Zeile 4000
2	Sprung nach Zeile 5000
3.8	Sprung nach Zeile 6000
4	Sprung nach Zeile 7000
5	Forts. des Programmlaufs in Zeile 440
6000	Forts. des Programmlaufs in Zeile 440

(2) In einem Programm stehende die folgenden Befehle:

```
  .
  .
500 ON GH GOTO 6000, 4000, 3000
510 PRINT "          STATISTIK"
  .
  .
```

Wert der Variablen GH	Auswirkung
6000	Forts. des Programmlaufs in Zeile 510
4000	Forts. des Programmlaufs in Zeile 510
3000	Forts. des Programmlaufs in Zeile 510
0	Forts. des Programmlaufs in Zeile 510
-1000	Forts. des Programmlaufs in Zeile 510
1	Sprung nach Zeile 6000
2	Sprung nach Zeile 4000
3	Sprung nach Zeile 3000
4	Forts. des Programmlaufs in Zeile 510

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 22.1** !

LE 22.2

Die Anweisung

ON .. GOSUB .. (von engl. "on", hier frei zu übersetzen mit "abhängig von"; GOSUB = Abkürzung für engl. "go to subprogram" = gehe zum Unterprogramm)

stellt eine Zusammenfassung von mehreren unmittelbar aufeinanderfolgenden IF .. THEN GOSUB-Konstruktionen dar (vgl. hierzu noch einmal **LE 18.2** / Seite 18-13: IF .. THEN-Anweisung / Format 2 !).

Beispiel für eine Mehrfachverzweigung ohne "ON .. GOSUB ..":

```
.
.
170 IF MA = 1 THEN GOSUB 3000
180 IF MA = 2 THEN GOSUB 4000
190 IF MA = 3 THEN GOSUB 5000
.
.
```

Die obigen IF .. THEN GOSUB-Anweisungen können wir in einem einzigen ON .. GOSUB-Befehl zusammenfassen:

Beispiel für eine Mehrfachverzweigung mit "ON .. GOSUB ..":

```
.
.
170 ON MA GOSUB 3000, 4000, 5000
.
.
```

Der Befehl könnte etwa folgendermaßen übersetzt werden:

Abhängig vom Inhalt der Variablen MA gehe zum jeweiligen Unterprogramm, das in Zeile 3000 oder 4000 oder 5000 beginnt (oder führe die nächste Anweisung aus)!

Ist A nämlich = 1, dann führt der Rechner jenes Unterprogramm aus, das bei der ersten nach GOSUB angegebenen Adresse (= 3000) beginnt; ist A = 2, dann führt der Rechner jenes Unterprogramm aus, das bei der zweiten nach GOSUB angegebenen Adresse (= 4000) beginnt; u. s. w.

Wir erkennen, daß der ON .. GOSUB-Befehl dem ON .. GOTO-Befehl sehr ähnlich ist. Der entscheidende Unterschied besteht darin, daß die nach GOSUB angegebenen Adressen jeweils den Anfang eines Unterprogramms darstellen müssen, jene nach GOTO sind hingegen beliebige Zeilennummern.

Im übrigen gelten die ergänzenden Hinweise, die wir beim ON .. GOTO-Befehl aufgeführt haben, beim ON .. GOSUB-Befehl in analoger Weise, nämlich

(1) Nach ON kann statt einer numerischen Variablen auch ein mathematischer Ausdruck (z. B. A * Z / 2) stehen.

(2) Durch den Wert des Ausdrucks bzw. der Variablen nach ON ist festgelegt, welches Unterprogramm auszuführen ist.

(3) Wenn das Ergebnis des Ausdrucks oder der Inhalt der angegebenen Variablen kein ganzzahliger Wert ist, dann erfolgt immer eine Abrundung zur Ganzzahl (z. B. 4.2 und 4.9 ergeben 4).

(4) Wenn der Wert des Ausdrucks / der Variablen nach ON größer ist als die Anzahl der angegebenen Adressen, dann wird der Befehl ausgeführt, der auf die ON .. GOSUB-Anweisung folgt.

(5) Wenn der Wert des Ausdrucks / der Variablen nach ON <= 0 ist, dann gelangt wie bei Pkt. (4) der Befehl zur Ausführung, der auf die ON .. GOSUB-Anweisung folgt. (Achtung: Insbesondere hier bestehen zwischen den verschiedenen BASIC-Dialekten erhebliche Unterschiede!)

PE 22.2

Versuchen Sie, zu der folgenden Aufgabe einen Plan zu erstellen, den Sie anschließend codieren sollen. (Hierfür stehen Ihnen die Leerseiten 22-13 bis 22-19 zur Verfügung!) Die Aufgabendefinition enthält absichtlich auch einige Codierungshinweise, die unbedingt zu berücksichtigen sind.

Falls Sie bei der Reihenfolgeplanung große Schwierigkeiten haben, ist es statthaft, die auf den Seiten 22-21 bis 22-24 aufgeführten Struktogramme zu konsultieren. Bei der Codierung sollten Sie sich aber ernsthaft um eine eigene Lösung bemühen.

Aufgabendefinition

Es soll berechnet werden, auf welchen Betrag ein beliebiges (einzugebendes) Anfangskapital (AK) in einer beliebigen (einzugebenden) Anzahl von Jahren (FJ) bei einem beliebigen (einzugebenden) Zinssatz (Z) anwächst.

Der Programmlauf ist zu beenden, wenn ein Anfangskapital von 0 eingegeben wird.

Abhängig vom Wunsch des Anwenders ist die Formel [1)]

(1) für einfache Verzinsung:
K1 = AK * (1 + Z * FJ / 100) ,

(2) für Zinseszins mit jährlichem Zuschlag:
K2 = AK * (1 + Z / 100) ↑ FJ ,

(3) für Zinseszins mit M-maligem Zinszuschlag / Jahr:
K3 = AK * (1 + Z / (M * 100)) ↑ (M * FJ) ,

(4) für stetige Verzinsung bei organischem Wachstum:
K4 = AK * EXP (Z * FJ / 100) [2)]

zu verwenden.

---

1) Bei der Wiedergabe der Formeln haben wir die BASIC-Schreibweise berücksichtigt. (Für die Lösung der Aufgabe ist das Verstehen der Formeln nicht erforderlich!)

2) Wie wir bereits in **LE 10** (vgl. Seite 10-02) gelernt haben, stellt BASIC zahlreiche vordefinierte Funktionen - wie z. B. SQR(X) - zur Verfügung. Eine weitere vordefinierte Funktion, die wir in dieser Formel verwenden, ist EXP(X). Mit EXP(X) kann man die Exponentialfunktion von X zur Basis e (d. h. e^X) berechnen (e = 2,718281828...).

Das Programm ist so zu konzipieren, daß die jeweilige Berechnung und Ergebnisausgabe in vier mit ON .. GOSUB .. aufzurufenden Unterprogrammen erfolgt. Da nicht nur das Endergebnis, sondern auch für jedes abgelaufene Jahr ein Zwischenergebnis auszugeben ist, sind in den jeweiligen Unterprogrammen die Berechnung und Ergebnisausgabe innerhalb einer FOR .. NEXT ..-Schleife durchzuführen, deren Endwert durch die Festlegungsdauer in Jahren (FJ) gebildet wird (also: .. FOR I = 1 TO FJ).

Nur in jenem Unterprogramm, das die Formel für Zinseszins mit M-maligem Zuschlag / Jahr enthält, muß vor der FOR .. NEXT ..-Schleife noch die Anzahl der gewünschten Zinszuschläge im Dialog ermittelt werden.

Nach der Ergebnisausgabe soll der Rücksprung ins Hauptprogramm erst dann erfolgen, wenn der Anwender die "W"-Taste ("W" von "weiter") gedrückt hat (= Warteschleife auf die "W"-Taste!).

Ergänzende Hinweise:

Die vier auf Seite 22-11 angegebenen Formeln beziehen sich auf ein zu berechnendes Endkapital nach FJ Jahren.

Gem. Aufgabenstellung ist nicht nur das Endergebnis, sondern auch für jedes abgelaufene Jahr ein Zwischenergebnis auszugeben. Deshalb sind die innerhalb der FOR .. NEXT ..-Schleife aufzuführenden Formeln folgendermaßen anzupassen:

(1) für einfache Verzinsung:

K1 = AK * (1 + Z * I / 100) ,

(2) für Zinseszins mit jährlichem Zuschlag:

K2 = AK * (1 + Z / 100) ↑ I ,

(3) für Zinseszins mit M-maligem Zuschlag / Jahr:

K3 = AK (1 + Z / (M * 100)) ↑ (M * I) ,

(4) für stetige Verzinsung bei organischem Wachstum:

K4 = AK * EXP (Z * I / 100) .

Struktogramme (1)

Struktogramme (2)

Struktogramme (3)

Struktogramme (4)

BASIC-Programm (1)

BASIC-Programm (2)

B A S I C - P r o g r a m m (3)

LÖS 22.2

Struktogramme zur **PE 22.2** :

Hinweise:

(1) In den folgenden Struktogrammen sind jene Befehle nicht berücksichtigt, die nur der optischen Verfeinerung dienen, also "Bildschirm löschen" (CLS) und "Leerzeile/n" (PRINT).

(2) Die Bedingung der Z-Schleife im Struktogramm auf Seite 22-21 haben wir im BASIC-Programm auf Seite 22-25 (vgl. Zeile 420) "umformuliert", um mit dem JA-Zweig der Bedingung die Befehle des Schleifenkörpers zu wiederholen. (Entsprechend den Regeln der Strukturierten Programmierung mußten wir im Struktogramm hingegen mit dem JA-Zweig die Z-Schleife verlassen.)

(3) In den bisherigen Struktogrammen hatten wir die Laufvariablen vor Beginn der Schleifen nicht auf 0 gesetzt, weil in BASIC alle numerischen Variablen automatisch den Anfangswert 0 erhalten. Aus logischer Sicht mußten wir jedoch in den Struktogrammen auf Seite 22-23 und 22-24 anders verfahren, da I u. U. mehrmals verwendet wird (Endekriterium für Programmlauf: AK = 0!). Gem. Aufgabenstellung wurde aber gefordert, die Schleifen mit FOR .. NEXT-Anweisungen zu codieren. Deshalb brauchen die Befehle I = 0 der Struktogramme nicht codiert zu werden, denn durch die FOR-Anweisung erhält I nach erneutem Aufruf des Unterprogramms immer wieder den angegebenen Anfangswert.

KAPEND

(UP-KONSTANTEN)

AUSGABE A1$

AUSGABE A2$

AUSGABE A3$

AUSGABE A4$

AUSGABE A5$

EINGABE AK

AK = 0

AUSGABE A6$;

EINGABE FJ

AUSGABE A7$;

EINGABE Z

AUSGABE A8$

AUSGABE A9$

AUSGABE B1$

AUSGABE B2$

AUSGABE B3$

AUSGABE B4$

AUSGABE B5$

AUSGABE B6$

EINGABE F

F = 1 OD. F = 2 OD. F = 3 OD. F = 4

F = ..

F=1: (UP-FORMEL-1)

F=2: (UP-FORMEL-2)

F=3: (UP-FORMEL-3)

F=4: (UP-FORMEL-4)

```
UP-KONSTANTEN
  A1$ = "DAS FOLGENDE PROGRAMM BERECHNET EIN"
  A2$ = "ENDKAPITAL MIT ZINSEN (INCL. DER"
  A3$ = "ZWISCHENERGEBNISSE FUER JEDES ABGELAU-"
  A4$ = "FENE JAHR)."
  A5$ = "ANFANGSKAPITAL"
  A6$ = "FESTLEGUNGSDAUER (IN JAHREN)"
  A7$ = "ZINSSATZ"
  A8$ = "ABHAENGIG VOM WUNSCH DES ANWENDERS WIRD"
  A9$ = "EINE DER FOLGENDEN FORMELN VERWENDET:"
  B1$ = "(1) EINFACHE VERZINSUNG"
  B2$ = "(2) ZINSESZINS MIT JAEHRLICHEM ZUSCHLAG"
  B3$ = "(3) ZINSESZINS MIT M-MALIGEM ZUSCHLAG/JAHR"
  B4$ = "(4) STETIGE VERZ. BEI ORG. WACHSTUM"
  B5$ = "WELCHE FORMEL SOLL VERWENDET WERDEN?"
  B6$ = "NUR 1 OD. 2 OD. 3 OD. 4 EINGEBEN!"
  B7$ = "ENDKAP. NACH"
  B8$ = " JAHR(EN) = DM"
  B9$ = "ANZAHL DER ZINSZUSCHLAEGE/JAHR"
  C1$ = "BITTE 'W'-TASTE DRUECKEN!"
```

```
UP-FORMEL-1
    I = 0
    I < FJ
        I = I + 1
        AUSGABE B7$; I; B8$; AK*(1+Z*I/100)
    (UP-WARTEN)
```

```
UP-FORMEL-2
    I = 0
    I < FJ
        I = I + 1
        AUSGABE B7$; I; B8$; AK*(1+Z/100)↑I
    (UP-WARTEN)
```

```
UP-FORMEL-3
    AUSGABE B9$;
    EINGABE M
    I = 0
    I < FJ
        I = I + 1
        AUSGABE B7$; I; B8$; AK*(1+Z/(M*100))↑(M*I)
    (UP-WARTEN)
```

```
UP-FORMEL-4
    I = 0
    I < FJ
        I = I + 1
        AUSGABE B7$; I; B8$; AK*EXP(Z*I/100)
    (UP-WARTEN)
```

```
UP-WARTEN
    AUSGABE TAB(12) C1$;
    WARTESCHLEIFE AUF "W"
```

BASIC-Programm zu **PE 22.2** :

```
10 REM **********
20 REM * KAPEND *
30 REM **********
40 GOSUB 1000
50 CLS
60 PRINT : PRINT : PRINT
70 PRINT A1$
80 PRINT
90 PRINT A2$
100 PRINT
110 PRINT A3$
120 PRINT
130 PRINT A4$
140 PRINT : PRINT : PRINT
150 PRINT A5$;
160 INPUT AK
170 PRINT
180 IF AK = 0 THEN 450
190 PRINT A6$;
200 INPUT FJ
210 PRINT
220 PRINT A7$;
230 INPUT Z
240 CLS
250 PRINT A8$
260 PRINT
270 PRINT A9$
280 PRINT
290 PRINT B1$
300 PRINT
310 PRINT B2$
320 PRINT
330 PRINT B3$
340 PRINT
350 PRINT B4$
360 PRINT : PRINT : PRINT
370 PRINT B5$
380 PRINT
390 PRINT B6$
400 PRINT
410 INPUT F
420 IF (F > 4) + (F < 1) THEN 380
```

```
430 ON F GOSUB 2000, 3000, 4000, 5000
440 GOTO 50
450 END
1000 REM ****************
1010 REM * UP-KONSTANTEN *
1020 REM ****************
1030 LET A1$ = "DAS FOLGENDE PROGRAMM BE
RECHNET EIN"
1040 LET A2$ = "ENDKAPITAL MIT ZINSEN (I
NCL. DER"
1050 LET A3$ = "ZWISCHENERGEBNISSE FUER
JEDES ABGELAU-"
1060 LET A4$ = "FENE JAHR)."
1070 LET A5$ = "ANFANGSKAPITAL"
1080 LET A6$ = "FESTLEGUNGSDAUER (IN JAH
REN)"
1090 LET A7$ = "ZINSSATZ"
1100 LET A8$ = "ABHAENGIG VOM WUNSCH DES
 ANWENDERS WIRD"
1110 LET A9$ = "EINE DER FOLGENDEN FORME
LN VERWENDET:"
1120 LET B1$ = "(1) EINFACHE VERZINSUNG"
1130 LET B2$ = "(2) ZINSESZINS MIT JAEHR
LICHEM ZUSCHLAG"
1140 LET B3$ = "(3) ZINSESZINS MIT M-MAL
. ZUSCHLAG/JAHR"
1150 LET B4$ = "(4) STETIGE VERZ. BEI OR
G. WACHSTUM"
1160 LET B5$ = "WELCHE FORMEL SOLL VERWE
NDET WERDEN?"
1170 LET B6$ = "NUR 1 OD. 2 OD. 3 OD. 4
EINGEBEN!"
1180 LET B7$ = "ENDKAP. NACH"
1190 LET B8$ = " JAHR(EN) = DM"
1200 LET B9$ = "ANZAHL DER ZINSZUSCHLAEG
E/JAHR"
1210 LET C1$ = "BITTE 'W'-TASTE DRUECKEN
!"
1220 RETURN
```

```
2000 REM **************
2010 REM * UP-FORMEL-1 *
2020 REM **************
2030 CLS
2040 PRINT : PRINT : PRINT
2050 FOR I = 1 TO FJ
2060 PRINT B7$; I; B8$; AK*(1+Z*I/100)
2070 NEXT I
2080 GOSUB 10000
2090 RETURN
3000 REM **************
3010 REM * UP-FORMEL-2 *
3020 REM **************
3030 CLS
3040 PRINT : PRINT : PRINT
3050 FOR I = 1 TO FJ
3060 PRINT B7$; I; B8$; AK*(1+Z/100)↑I
3070 NEXT I
3080 GOSUB 10000
3090 RETURN
4000 REM **************
4010 REM * UP-FORMEL-3 *
4020 REM **************
4030 CLS
4040 PRINT : PRINT : PRINT
4050 PRINT B9$;
4060 INPUT M
4070 PRINT
4080 FOR I = 1 TO FJ
4090 PRINT B7$; I; B8$; AK*(1+Z/(M*100))
↑(M*I)
4100 NEXT I
4110 GOSUB 10000
4120 RETURN
5000 REM **************
5010 REM * UP-FORMEL-4 *
5020 REM **************
5030 CLS
5040 PRINT : PRINT : PRINT
5050 FOR I = 1 TO FJ
5060 PRINT B7$; I; B8$; AK*EXP(Z*I/100)
5070 NEXT I
5080 GOSUB 10000
5090 RETURN
```

```
10000 REM ************
10010 REM * UP-WARTEN *
10020 REM ************
10030 PRINT : PRINT : PRINT : PRINT
10040 PRINT TAB(12) C1$;
10050 GET A$
10060 IF A$ <> "W" THEN 10050
10070 RETURN
```

Es wäre sehr ungewöhnlich, wenn die von Ihnen erarbeitete Lösung mit der unter LÖS 22.2 wiedergegebenen vollkommen übereinstimmt. Sie sollten daher vor allem überprüfen, ob Ihr Programm funktionsfähig ist und ob Sie den ON .. GOSUB ..-Befehl richtig verwendet haben.

Zur besseren Orientierung finden Sie auf den beiden folgenden Seiten 22-29 und 22-30 die Ergebnisse eines Testlaufs zum obigen Programm!

Ergebnis eines Testlaufs zum Programm aus **LÖS 22.2** :

```
RUN
```

```
DAS FOLGENDE PROGRAMM BERECHNET EIN
ENDKAPITAL MIT ZINSEN (INCL. DER
ZWISCHENERGEBNISSE FUER JEDES ABGELAU-
FENE JAHR).

ANFANGSKAPITAL? 100000
FESTLEGUNGSDAUER (IN JAHREN)? 5
ZINSSATZ? 8
```

```
ABHAENGIG VOM WUNSCH DES ANWENDERS WIRD
EINE DER FOLGENDEN FORMELN VERWENDET:
(1) EINFACHE VERZINSUNG
(2) ZINSESZINS MIT JAEHRLICHEM ZUSCHLAG
(3) ZINSESZINS MIT M-MAL. ZUSCHLAG/JAHR
(4) STETIGE VERZ. BEI ORG. WACHSTUM

WELCHE FORMEL SOLL VERWENDET WERDEN?
NUR 1 OD. 2 OD. 3 OD. 4 EINGEBEN!
? 2
```

```
ENDKAP. NACH 1 JAHR(EN) = DM 108000
ENDKAP. NACH 2 JAHR(EN) = DM 116640
ENDKAP. NACH 3 JAHR(EN) = DM 125971.2
ENDKAP. NACH 4 JAHR(EN) = DM 136048.9
ENDKAP. NACH 5 JAHR(EN) = DM 146932.81


            BITTE 'W'-TASTE DRUECKEN!
```

```
DAS FOLGENDE PROGRAMM BERECHNET EIN

ENDKAPITAL MIT ZINSEN (INCL. DER

ZWISCHENERGEBNISSE FUER JEDES ABGELAU-

FENE JAHR).


ANFANGSKAPITAL? 0

Ready
```

LE 22.3

In **LE 15.1** (vgl. Seite 15-02 f., Pkt. (2)) hatten wir gelernt, daß bei der Eingabe eines unzulässigen Wertes (String statt einer Zahl) im Rahmen des INPUT-Befehls das System die Verarbeitung mit folgender Fehlermeldung unterbricht:

```
Illegal data error in ..

Ready
```

(frei übersetzt: Fehler wegen unzulässiger Daten in Zeile .. - fertig)

Beispiel:

Der folgende Programmausschnitt

```
 .
 .
 .
140 PRINT "ANFANGSKAPITAL";
150 INPUT AK
 .
 .
 .
```

führt bei fehlerhafter Eingabe zu folgendem Ergebnis:

```
 .
 .
 .
ANFANGSKAPITAL? 100000 DM
Illegal data error in 150
Ready
```

Die Eingabe (Zeile 150) ist fehlerhaft, weil in der numerischen Variablen AK ein String (1000_DM) nicht gespeichert werden kann.

Eine Programmunterbrechung mit Fehlermeldung kann man umgehen, wenn man v o r dem Befehl, der möglicherweise eine Unterbrechung verursacht, die Anweisung

ON ERROR GOTO ..	(von engl. "on", hier frei zu übersetzen mit "bei Vorliegen (eines)" oder einfach mit "bei"; von engl. "error" = Fehler; von engl. "to go to" = gehen nach)

erteilt. Der Befehl könnte demnach etwa folgendermaßen übersetzt werden:

"Bei Vorliegen eines Fehlers gehe nach .. "

In der Subroutine, die mit jener Zeilennummer beginnt, die nach ON ERROR GOTO aufgeführt ist, sollte man eine Fehlererläuterung ausgeben, um dann mit dem Befehl

RESUME ..	(von engl. "to resume" = wiederaufnehmen, fortsetzen)

festzulegen, an welcher Stelle des Programms die Verarbeitung wiederaufzunehmen ist, und zwar bestehen folgende Möglichkeiten:

RESUME	Rückkehr zur fehlerverursachenden Zeile,
RESUME NEXT	Fortsetzung des Programms mit jener Zeile, die auf die fehlerverursachende folgt,
RESUME Zeilennummer	Fortsetzung des Programms mit der angegebenen Zeilennummer,
RESUME 0	Neustart des Programms.

Unseren Programmausschnitt von Seite 22-31 können wir also folgendermaßen ergänzen:

Beispiel (mit ON ERROR GOTO ../RESUME ..):

```
 .
 .
 .
140 PRINT "ANFANGSKAPITAL";
145 ON ERROR GOTO 1000                  <—— ergänzter Befehl
150 INPUT AK
 .
 .
 .
300 END

1000 REM FEHLERBEHANDLUNGSROUTINE     |
1010 REM -----------------------      |
1020 PRINT "EINGABEFEHLER! - ";        |<—— ergänzte Subroutine
1030 PRINT "EINGABE WIEDERHOLEN!"      |
1040 RESUME 140                        |


RUN
 .
 .
 .
ANFANGSKAPITAL? 100000 DM
EINGABEFEHLER! - EINGABE WIEDERHOLEN!
ANFANGSKAPITAL?
```

PE 22.3

Bei der Arbeit mit einem BASIC-Programm kommt es nur an einer bestimmten Stelle immer wieder zu einer Programmunterbrechung mit der folgenden Fehlermeldung:

```
.
.
.
Over flow error in 450
Ready
```

An der fehleranfälligen Programmstelle stehen die folgenden Befehle:

```
.
.
.
400 PRINT "GESAMTKOSTEN DER REISE";
410 INPUT GK
420 PRINT
430 PRINT "ANZAHL DER TEILNEHMER";
440 INPUT T
450 LET D = GK / T
.
.
.
```

Bei der Untersuchung der Ursache wurde festgestellt, daß die in der ADV unerfahrenen Anwender als Teilnehmerzahl nicht selten den Wert 0 eingeben. Hierdurch entsteht in Zeile 450 der Programmabbruch mit der obigen Fehlermeldung, weil in BASIC eine Division durch 0

unzulässig ist.

Ergänzen Sie das Programm dahingehend, daß eine fehlerhafte Eingabe an der angegebenen Stelle nicht mehr zu einer Programmunterbrechung führt. In der Fehlerbehandlungsroutine (ab Zeile 2000) soll für den Anwender ein erläuternder Hinweis ausgegeben werden.

```
.
.
.
400 PRINT "GESAMTKOSTEN DER REISE";
410 INPUT GK
420 PRINT
430 PRINT "ANZAHL DER TEILNEHMER";
440 INPUT T

445 .......................................
450 LET D = GK / T
.
.
.
1800 END
2000 REM FEHLERBEHANDLUNGSROUTINE
2010 REM ------------------------

2020 ......................................

2030 ......................................

2040 ......................................
```

LÖS 22.3

Programmausschnitt aus **PE 22.3** mit den Ergänzungen:

```
.
.
.
400 PRINT "GESAMTKOSTEN DER REISE";
410 INPUT GK
420 PRINT
430 PRINT "ANZAHL DER TEILNEHMER";
440 INPUT T
445 ON ERROR GOTO 2000
450 LET D = GK / T
.
.
.
1800 END
2000 REM FEHLERBEHANDLUNGSROUTINE
2010 REM ------------------------
2020 PRINT "UNZULAESSIGE EINGABE!"
2030 PRINT "DESHALB NOCH EINMAL:"
2040 RESUME 430
```

Sollte Ihre Lösung nicht der obigen entsprechen, kehren Sie zurück zur
→ **LE 22.3** !

LE 22.4

Die folgenden Ausführungen sind speziellen Details bei der Fehlerbehandlung gewidmet und wenden sich daher eher an den erfahrenen Programmierer. Der Anfänger sollte diesen Abschnitt zunächst übergehen. Setzen Sie Ihre Arbeit also ggf. mit Lektion 23 (Seite 23-01) fort!

Bei der Fehlerbehandlung mit dem O N E R R O R G O T O ..-Befehl sind folgende Einzelheiten zu berücksichtigen:

(1) Nach einem O N E R R O R G O T O ..-Befehl mit angegebener Zeilennummer wird bei Auftreten eines Fehlers zu der angegebenen Zeile verzweigt. Hier beginnt dann überlicherweise - wie wir es in der vorangegangenen **LE** erläutert haben - die vom Programmierer erstellte Fehlerbehandlungsroutine.

(2) Nach einem O N E R R O R G O T O ..-Befehl mit angegebener Zeilennummer werden alle folgenden Fehler an dieselbe Fehlerbehandlungsroutine verwiesen, was häufig nicht erwünscht ist. Mit der Anweisung

ON ERROR GOTO 0

kann man aber einen vorangegangenen Befehl O N E R R O R G O T O (m i t Z e i l e n n u m m e r) wieder neutralisieren. Fehler, die nach einem O N E R R O R G O T O 0 auftreten, verursachen dann wieder eine Programmunterbrechung mit der jeweiligen Fehlermeldung des Systems.

(3) Wenn der Befehl O N E R R O R G O T O 0 in der Fehlerbehandlungsroutine erscheint, dann erfolgt dort eine Programmunterbrechung mit der jeweiligen Fehlermeldung des Systems.

Hiervon sollte man Gebrauch machen, wenn ein Fehler auftritt, der durch eine Fehlerbehandlungsroutine nicht aufgefangen werden kann. (Vgl. hierzu auch die folgenden Ausführungen zu den System-Variablen ERN und ERL.)

(4) Wenn zusätzlich in der Fehlerbehandlungsroutine, die aufgrund eines ON ERROR GOTO ..-Befehls angesprungen wurde, ein Fehler vorliegt, dann erfolgt eine Programmunterbrechung mit der jeweiligen Fehlermeldung des Systems.

ERN / ERL

ERN und ERL gehören zu den reservierten BASIC-Wörtern, und zwar sind es System-Variablen (vgl. Seite 8-16 ff.).

Wenn aufgrund eines ON ERROR GOTO ..-Befehls bei Auftreten eines Fehlers zu der Fehlerbehandlungsroutine verzweigt wurde, dann enthält die System-Variable

ERN (Abkürzung für engl. "error number" = Fehlernummer)

die jeweilige im System festgelegte Fehlermeldungsnummer[1], und

ERL (Abkürzung für engl. "error line, frei übersetzt etwa "fehlerverursachende Zeile")

enthält die Zeilennummer, in der der Fehler auftritt.

Mit Hilfe der Variablen ERN und ERL ist nun eine differenzierte Fehlerbehandlung in der Fehlerbehandlungsroutine möglich.

Eine differenzierte Fehlerbehandlung nach der Art des Fehlers geschieht durch Abfrage des Inhalts von ERN mit Hilfe von IF .. THEN ..-Anweisungen, beispielsweise

---

1) Vgl. hierzu Anhang-33 bis Anhang-39 (Systemfehlermeldungen)

```
.
.
.
10000 IF ERN = 2 THEN 15000
10010 IF ERN = 3 THEN 16000
10020 IF ERN = 43 THEN 17000
.
.
.
```

Eine differenzierte Fehlerbehandlung nach dem Ort (= Zeilennummer) des Fehlerauftretens geschieht durch Abfrage des Inhalts von ERL mit Hilfe von IF .. THEN ..-Anweisungen, beispielsweise

```
.
.
.
20000 IF ERL = 200 THEN 20100
20010 IF ERL = 370 THEN 20200
20020 IF ERL = 490 THEN 20300
.
.
.
```

Bei der Untersuchung des Inhalts von ERL sollte - wie im obigen Beispiel gezeigt - die Zeilennummer immer rechts vom Gleichheitszeichen stehen (also nicht so: .. IF 200 = ERL), weil andernfalls bei Verwendung des Systemkommandos RENUM eine Neunumerierung nicht möglich ist.

Außerdem ist zu beachten, daß es nicht zulässig ist, den System-Variablen ERN und ERL im Rahmen einer LET-Anweisung einen Wert zuzuweisen.

Nur aus Gründen der Vollständigkeit wollen wir abschließend noch die Anweisung

ERROR .. (von engl. "error", hier zu übersetzen mit "Fehler")

behandeln, obwohl sie in der üblichen BASIC-Programmierung keine besondere Bedeutung besitzt.

Der ERROR ..-Befehl kann für zwei Zwecke eingesetzt werden.

(1) Mit Hilfe dieser Anweisung hat der Programmierer die Möglichkeit, den Programmlauf zu unterbrechen und eine Systemfehlermeldung ausgeben zu lassen, ohne daß der ausgegebene Systemfehler vorliegt. Die Ganzzahl nach ERROR ist die festgelegte Fehlermeldungsnummer (vgl. hierzu Anh.-33 - Anh.-39!)

Beispiel:

```
10 PRINT "WIE HEISST DER AUTOR DES"
20 PRINT "BUCHES, DAS SIE Z. ZT."
30 PRINT "BEARBEITEN?"
40 INPUT A$
50 IF A$ <> "HAMANN" THEN ERROR 3
60 PRINT "SIE BESITZEN EIN GUTES GEDAECH
TNIS!"
70 END
RUN
WIE HEISST DER AUTOR DES
BUCHES, DAS SIE Z. ZT.
BEARBEITEN?
? PUPECKA
Illegal data error in 50
Ready
```

Die Fehlermeldung (Illegal data error ..) mit der im System festgelegten Nummer 3 wird also im obigen Beispiel duch den Befehl "... ERROR 3" für programmspezifische Zwecke eingesetzt.

(2) Mit Hilfe dieser Anweisung hat der Programmierer die Möglichkeit, eigene Fehlermeldungen zu generieren.

Beispiel:

```
 .
 .
 .
450 ON ERROR GOTO 10000
460 PRINT "GRUNDPREIS";
470 INPUT P
480 IF P > 50000 THEN ERROR 230
 .
 .
 .
900 END
 .
 .
 .
10000 IF ERN = 230 THEN PRINT "UNSINNIGE
 ANGABE!"
10010 IF ERL = 480 THEN RESUME 460
```

Es ist darauf zu achten, daß eine Ganzzahl im Bereich von 0 - 255 verwendet wird, die nicht für eine Systemfehlermeldung belegt ist.

PE 22.4

(1) Zu Beginn eines Programmabschnitts steht der folgende Befehl:

```
600 ON ERROR GOTO 12000
```

In einem späteren Programmabschnitt möchten Sie den obigen Befehl neutralisieren. Wie lautet die entsprechende Anweisung ?

700 ....

(2) Was steht nach Auftreten eines Fehlers in den System-Variablen ERN und ERL ?

in ERN: ....

in ERL: ....

(3) In einer Fehlerbehandlungsroutine wollen Sie nach Ausgabe von bestimmten Erläuterungen veranlassen, daß die Verarbeitung in der Befehlszeile 880 wieder aufgenommen wird. Wie lautet die Anweisung ?

2020 ....

LÖS 22.4

(1) Der Befehl

600 ON ERROR GOTO 12000

kann mit der folgenden Anweisung neutralisiert werden:

700 ON ERROR GOTO 0 .

(2) Nach Auftreten eines Fehlers steht

in ERN: die jeweilige im System festgelegte Fehlermeldungsnummer,

in ERL: die Zeilennummer, in der sich der Fehler befindet.

(3) Die Anweisung in einer Fehlerbehandlungsroutine, mit der man veranlaßt, daß die Verarbeitung in der Befehlszeile 880 wieder aufgenommen wird, lautet

2020 RESUME 880 .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 22.4** !

ÜBUNGsprogramm zu Lektion 22

Problemstellung:

Im Lukas-Evangelium, Kapitel 20, Vers 20 - 26, wird folgendes berichtet (vgl. auch Matthäus 22, 15 - 22 und Markus 12, 13 - 17):

20. Und sie stellten ihm nach und sandten Leute aus, die sich stellen sollten, als wären sie fromm, auf daß sie ihn in seiner Rede fingen, damit sie ihn überantworten könnten der Obrigkeit und Gewalt des Landpflegers.

21. Und sie fragten ihn und sprachen: Meister, wir wissen, daß du aufrichtig redest und lehrest und achtest keines Menschen Ansehen, sondern du lehrest den Weg Gottes recht.

22. Ist's recht, daß wir dem Kaiser Steuer geben, oder nicht?

23. Er aber merkte ihre List und sprach zu ihnen:

24. Zeiget mir einen Groschen! Wes Bild und Aufschrift hat er? Sie aber sprachen: Des Kaisers.

25. Er aber sprach zu ihnen: So gebet dem Kaiser, was des Kaisers ist, und Gott, was Gottes ist!

26. Und sie konnten ihn nicht fassen bei dem Wort vor dem Volk und verwunderten sich seiner Antwort und schwiegen stille.

Es kann davon ausgegangen werden, daß die obige Begebenheit 30 n. Chr. stattgefunden hat. Bei dem Geldstück, das in der (neu bearbeiteten) Luther-Übersetzung mit "Groschen" bezeichnet wird, handelte es sich um einen Denar[1].

---

1) Denar = römische Silbermünze von 3,89 g
Nach Ansicht von Numismatikern muß es sich um ein Stück gehandelt haben, das auf der Vorderdseite den belorbeerten Kopf des Kaisers Tiberius mit der Inschrift TI CAESAR DIVI AUG F AUGUSTUS und auf der Rückseite die sitzende Livia (Mutter des Herrschers) mit dem Text PONIF MAXIM wiedergibt. Ein gut erhaltenes Exemplar dieser Münze hat heute einen Marktwert von ca. DM 1.000,—.

Programmbeschreibung (Jesus und die Verzinsung des Denars):

Das folgende Programm berechnet, auf welchen Betrag der Denar bis heute angewachsen wäre, wenn Jesus das Geldstück damals (30 n. Chr.), im Anschluß an die geschilderte Begebenheit, auf einer Bank verzinslich angelegt hätte.

(Für die Berechnung gehen wir davon aus, daß 1 Denar dem Wert 1 DM entspricht; das Inflationsproblem und zwischenzeitliche, widrige politische Umstände werden vernachlässigt.)

Das Programm ist so konzipiert, daß zunächst alle benötigten Zeichenkettenkonstanten String-Variablen zugewiesen werden (im Unterprogramm Vorlauf), dann erscheint der Bericht und die Erläuterung des Programms (2 Bilder durch das Unterprogramm Bericht), und anschließend ist der gewünschte Zinssatz und das laufende Jahr einzugeben (im Unterprogramm Eingabe). Nach Berechnung des Endkapitals mit der Zinseszinsformel (im Unterprogramm Berechnung) erfolgt die Ausgabe des Ergebnisses (im Unterprogramm Ausgabe). Ein OVERFLOW ERROR, der bei Eingabe eines zu hohen Zinssatzes auftritt, wird mit einer Fehlerbehandlungsroutine aufgefangen.

Programmliste:

```
10 REM *************************
20 REM *     JESUS UND DIE     *
30 REM * VERZINSUNG DES DENARS *
40 REM *************************
50 REM
60 REM HAUPTPROGRAMM:
70 REM -------------
80 GOSUB 1000  : REM UP-VORLAUF
90 GOSUB 2000  : REM UP-BERICHT
100 GOSUB 3000 : REM UP-EINGABE
110 GOSUB 4000 : REM UP-BERECHNUNG
120 GOSUB 5000 : REM UP-AUSGABE
130 END
970 REM UNTERPROGRAMME:
980 REM --------------
990 REM
```

```
1000 REM UP-VORLAUF
1010 REM ==========
1020 LET A1$ = " IM LUKAS-EVANGELIUM, KA
PITEL 20,"
1030 LET A2$ = " VERS 20 - 26, WIRD BERI
CHTET, DASS"
1040 LET A3$ = " JESUS AUF DIE FRAGE, OB
 ES RECHTENS"
1050 LET A4$ = " SEI, DASS MAN DEM KAISE
R STEUERN"
1060 LET A6$ = " ZAHLEN MUESSE, SICH EIN
EN DENAR"
1070 LET A7$ = " HABE GEBEN LASSEN UND D
ANN FOLGENDES"
1080 LET A8$ = " GEANTWORTET HABE:"
1090 LET A9$ = " SO GEBET DEM KAISER, WA
S DES KAISERS"
1100 LET B1$ = " IST, UND GOTT, WAS GOTT
ES IST!"
1110 LET B2$ = " DIESES PRGRAMM BERECHNE
T, AUF WELCHEN"
1120 LET B3$ = " BETRAG DAS GELD BEI VER
ZINSLICHER"
1130 LET B4$ = " ANLAGE ANGEWACHSEN WAER
E."
1140 LET B5$ = " (FUER DIE BERECHNUNG WI
RD UNTERSTELLT,"
1150 LET B6$ = " DASS 1 DENAR DEM WERT 1
 DM ENTSPRICHT."
1160 LET B7$ = " DAS INFLATIONSPROBLEM U
ND ZWISCHEN-"
1170 LET B8$ = " ZEITLICHE, WIDRIGE POLI
TISCHE UM-"
1180 LET B9$ = " STAENDE WERDEN VERNACHL
AESSIGT.)"
1190 LET C1$ = " 'W'-TASTE DRUECKEN!"
1200 LET C2$ = " ZINSSATZ "
1210 LET C3$ = " LFD. JAHR"
1220 LET C4$ = " BEI EINEM ZINSSATZ VON"
1230 LET C6$ = " BELIEFE SICH DAS ENDKAP
ITAL IM"
1240 LET C5$ = " %"
1250 LET C7$ = " JAHRE"
1260 LET C8$ = " AUF"
```

```
1270 LET C9$ = " DM."
1280 LET D1$ = " EINE BERECHNUNG IST NIC
HT MOEGLICH,"
1290 LET D2$ = " DA AUFGRUND DER HOEHE D
ES ƷINSES"
1300 LET D3$ = " EIN OVERFLOW ERROR ENTS
TEHT!"
1310 LET D4$ = " BITTE GEBEN SIE EINEN N
IEDRIGEREN"
1320 LET D5$ = " ƷINSSATƷ EIN!"
1330 RETURN
2000 REM UP-BERICHT
2010 REM ==========
2020 CLS
2030 PRINT : PRINT
2040 PRINT A1$
2050 PRINT
2060 PRINT A2$
2070 PRINT
2080 PRINT A3$
2090 PRINT
2100 PRINT A4$
2110 PRINT
2120 PRINT A6$
2130 PRINT
2140 PRINT A7$
2150 PRINT
2160 PRINT A8$
2170 PRINT : PRINT
2180 PRINT A9$
2190 PRINT
2200 PRINT B1$
2210 PRINT : PRINT
2220 PRINT TAB(20) C1$
2230 GET A$ : IF A$ <> "W" THEN 2230
2240 CLS
2250 PRINT : PRINT
2260 PRINT B2$
2270 PRINT
2280 PRINT B3$
2290 PRINT
2300 PRINT B4$
2310 PRINT : PRINT
2320 PRINT B5$
```

```
2330 PRINT
2340 PRINT B6$
2350 PRINT
2360 PRINT B7$
2370 PRINT
2380 PRINT B8$
2390 PRINT
2400 PRINT B9$
2410 PRINT : PRINT : PRINT : PRINT
2420 PRINT TAB(20) C1$
2430 GET A$ : IF A$ <> "W" THEN 2430
2440 RETURN
3000 REM UP-EINGABE
3010 REM ==========
3020 CLS
3030 PRINT : PRINT : PRINT
3040 PRINT C2$;
3050 INPUT Z
3060 PRINT : PRINT : PRINT
3070 PRINT C3$;
3080 INPUT LJ
3090 RETURN
4000 REM UP-BERECHNUNG
4010 REM =============
4020 ON ERROR GOTO 5150
4030 LET EK = 1 * (1+ Z/100) ↑ (LJ-30)
4040 RETURN
5000 REM UP-AUSGABE
5010 REM ==========
5020 CLS
5030 PRINT : PRINT
5040 PRINT C4$
5050 PRINT
5060 PRINT Z; C5$
5070 PRINT
5080 PRINT C6$
5090 PRINT
5100 PRINT C7$; LJ; C8$
5110 PRINT
5120 PRINT EK; C9$
5130 PRINT : PRINT : PRINT
5140 RETURN
5150 REM
```

```
10000 REM SUBROUTINE ZUR
10010 REM FEHLERBEHANDLUNG
10020 REM ================
10030 IF (ERN=2)*(ERL=4030) THEN 10050
10040 ON ERROR GOTO 0
10050 CLS
10060 PRINT : PRINT : PRINT
10070 PRINT D1$
10080 PRINT
10090 PRINT D2$
10100 PRINT
10110 PRINT D3$
10120 PRINT : PRINT : PRINT
10130 PRINT D4$
10140 PRINT
10150 PRINT D5$
10160 PRINT : PRINT : PRINT
10170 PRINT TAB(20) C1$
10180 GET A$ : IF A$ <> "W" THEN 10180
10190 RESUME 100
```

23. Feldreservierung mit DIM / Indizierte Variablen

LE 23.1

In Lektion 8 hatten wir die folgenden Variablentypen kennengelernt:

		Beispiele:			
(1)	Numerische Variablen	A	Z3	GH	PO
(2)	String-Variablen	A$	A1$	DU$	TX$
(3)	System-Variablen	ERN	ERL	TI$	SIZE

Nun besteht in BASIC die Möglichkeit, mit numerischen Variablen und mit String-Variablen sog. Felder (engl. "arrays") zu bilden. Das bedeutet, daß unter einem Variablennamen mehrere Datenfelder angelegt werden, die man nur durch eine laufende Nummer (= Index) unterscheidet: Man spricht daher auch von "indizierten Variablen".

Beispiel:

175	218	13	7	99
A(0)	A(1)	A(2)	A(3)	A(4)

Venn man eine bestimmte Variable des obigen Feldes ansprechen ıöchte, dann ist die Nennung nur des Variablennamens A nicht usreichend, weil alle fünf Feldelemente den gleichen Namen ıämlich A) haben. Zusätzlich muß man noch einen Index (in lammern hinter dem Variablennamen) angeben.

Will man beim SHARP MZ-700 indizierte Variablen verwenden, so ist grundsätzlich vorher mit

DIM .. (von engl. "dimension" = Dimension)

zu bestimmen, wieviel "Plätze" benötigt werden. Beispielsweise reserviert die Anweisung

```
.
.
80 DIM B(12)
.
.
```

die folgenden 13 Feldelemente - also nicht etwa 12, denn das System beginnt immer mit dem Index 0 - die außerdem automatisch den Anfangswert 0 erhalten:

0	0	0	0	0	0	0	0	0
B(0)	B(1)	B(2)	B(3)	B(4)	B(5)	B(6)	B(7)	B(8)

0	0	0	0
B(9)	B(10)	B(11)	B(12)

MERKE:

Ohne vorangegangene Felddefinition mit D I M darf man beim SHARP MZ-700 indizierte Variablen nicht benutzen.[1)]

1) In einigen BASIC-Dialekten ist es zulässig, indizierte Variablen zu benutzen, ohne daß vorher eine DIM-Anweisung erteilt wurde. Das System reserviert dann automatisch Felder mit einem Indexmaximalwert von 10. Das hat meistens zur Folge, daß entweder zu viele Datenfelder reserviert werden (dadurch Arbeitsspeicherplatzverschwendung) oder aber zu wenig zur Verfügung stehen würden, so daß der Programmierer dann doch auf die DIM-Anweisung zurückgreifen muß.

Nachdem man mit DIM ein Feld reserviert hat, bestehen für die Indizierung mehrere Möglichkeiten:

(1) Index als Ganzzahl

Beispiel:

```
10 DIM B(14)
 .
 .
 .
70 LET B(5) = 175
 .
 .
 .
```

(2) Index als numerische Variable

Beispiel:

```
10 INPUT "ANZAHL DER TEILNEHMER? "; T
20 DIM TN$(T)
 .
 .
 .
170 INPUT "GEWUENSCHTES FELD? "; F
180 PRINT TN$(F)
 .
 .
 .
```

(3) Index als indizierte Variable

B e i s p i e l :

```
10 DIM MO$(11)
20 DIM M(11)
30 FOR I = 0 TO 11
40 LET M(I) = I
50 NEXT I
 .
 .
 .
190 LET MO$(M(0)) = "JANUAR"
 .
 .
 .
```

(4) Index als mathematischer Ausdruck

B e i s p i e l :

```
10 DIM MO$(11)
20 FOR I = 0 TO 11
30 READ MO$(I)
40 NEXT I
 .
 .
 .
150 INPUT "WELCHER MONAT (1-12)? "; NR
160 PRINT "DER MONAT LAUTET ";
170 PRINT MO$(NR-1)
 .
 .
 .
300 END
500 DATA JANUAR, FEBRUAR, MAERZ
510 DATA APRIL, MAI, JUNI
520 DATA JULI, AUGUST, SEPTEMBER
530 DATA OKTOBER, NOVEMBER, DEZEMBER
```

Ergänzende Hinweise:

(1) Für die Namensbildung gelten analog die gleichen Regeln wie für "normale" Variablen (vgl. hierzu die Ausführungen auf Seite 8-09 und 8-15, Pkt. (1) und (2)).

(2) Ist der Wert des Index nicht ganzzahlig, dann erfolgt immer eine Abrundung zur Ganzzahl.

(3) Der Hinweis von Seite 8-10 soll hier nochmals wiederholt werden, weil er auch für Felder gilt:

Anders als andere Programmiersprachen setzt BASIC die numerischen Variablen zu Beginn des Programmlaufs auf 0 (Null). Und wird der Inhalt einer String-Variablen ausgegeben, bevor ihr Zeichen zugewiesen worden sind, erscheinen nur Leerstellen.

PE 23.1

Mit dem folgenden Programm soll der durchschnittliche Tagesumsatz einer Unternehmung in einer bestimmten Arbeitswoche ermittelt werden. Die Tagesumsätze sind in ein Feld A einzugeben.

Ergänzen Sie die indizierte Variable in den Zeilen 120 und 130!

```
10 REM ***************
20 REM * INDIZIERUNG *
30 REM ***************
40 CLS
50 PRINT "ANZAHL DER ARBEITSTAGE";
60 INPUT X
70 DIM A(X-1)
80 PRINT
90 PRINT "TAGESUMSAETZE DER WOCHE EINGEB
EN!"
100 PRINT "(1 WERT PRO ZEILE!)"
110 FOR N = 0 TO X-1

120 INPUT ....................

130 LET S = S + ....................
140 NEXT N
150 PRINT
160 PRINT "DURCHSCHNITTLICHER UMSATZ PRO
 TAG:"
170 PRINT
180 PRINT S/X; " DM"
190 END
```

LÖS 23.1

Programm aus **PE 23.1** mit den ergänzten indizierten Variablen:

```
10 REM ***************
20 REM * INDIZIERUNG *
30 REM ***************
40 CLS
50 PRINT "ANZAHL DER ARBEITSTAGE";
60 INPUT X
70 DIM A(X-1)
80 PRINT
90 PRINT "TAGESUMSAETZE DER WOCHE EINGEB
EN!"
100 PRINT "(1 WERT PRO ZEILE!)"
110 FOR N = 0 TO X-1
120 INPUT A(N)
130 LET S = S + A(N)
140 NEXT N
150 PRINT
160 PRINT "DURCHSCHNITTLICHER UMSATZ PRO
 TAG:"
170 PRINT
180 PRINT S/X; " DM"
190 END
```

Sollte Ihre Lösung nicht richtig sein, kehren Sie zurück zur ➡ **LE 23.1** !

LE 23.2

Die in der vorangegangenen **LE 23.1** vorgestellten Felder waren eindimensional, d. h. die indizierten Variablen konnten in einer Zeile dargestellt werden. (Ein eindimensionales Feld wird bisweilen kurz "Liste" oder "Vektor" genannt.)

BASIC ermöglicht aber auch die Definition und Verwendung von zweidimensionalen Feldern[1]. (Ein zweidimensionales Feld bezeichnet man außerdem als "Tabelle" oder "Matrix".) Beispielsweise reserviert die Anweisung

```
100 DIM A(3, 4)
```

eine Matrix, die aus 20 "Unterfeldern" besteht:

S p a l t e n

A	0	1	2	3	4
Zeilen → 0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

1) Bei nicht wenigen BASIC-Systemen sind hiermit die Grenzen hinsichtlich der Anzahl der zulässigen Dimensionen erreicht. Beim SHARP MZ-700 können hingegen sogar bis maximal 4 Dimensionen definiert werden.

Wenn man einen bestimmten Platz in der Matrix ansprechen will, dann müssen zusätzlich zum Variablennamen zwei Indizes (in Klammern hinter dem Variablennamen, durch ein Komma getrennt) angegeben werden:

- der erste Index für die Zeilennummer,
- der zweite Index für die Spaltennummer.

Mit diesen beiden Angaben ist jeder Platz genau beschrieben.

Will man z. B. dem stark umrandeten Feld in der Darstellung auf Seite 23-09 den Wert 167 zuweisen, so könnte man dies mit der folgenden Anweisung bewirken:

```
.. LET A(2, 4) = 167
```

Zeilennummer (2) — Spaltennummer (4)

Für das Füllen einer Tabelle mit bestimmten Werten (bzw. für deren Ausgabe) eignen sich sehr häufig zwei ineinandergeschachtelte FOR .. NEXT ..-Schleifen.

Beispiel:

```
10 REM ****************
20 REM * UMSATZZAHLEN *
30 REM ****************
40 CLS
50 DIM T(4, 3)
60 PRINT "EINGABE VON SOLL-UMSATZZAHLEN
(STUECK)"
70 PRINT
80 PRINT "PRO QUARTAL FUER FUENF ARTIKEL
GRUPPEN:"
90 FOR A = 0 TO 4
100 PRINT
110 PRINT "UMSATZ FUER ARTIKEL"; A+1
```

```
120 FOR Q = 0 TO 3
130 PRINT "IN QUARTAL"; Q+1
140 INPUT T(A, Q)
150 NEXT Q
160 NEXT A
 .
 .
 .
```

PE 23.2

(1) Ein Programm beginnt mit dem folgenden Befehl:

```
10 DIM T(5, 6)
```

Wieviel Elemente werden durch diese Anweisung reserviert?

. . . .

(2) Wenn man einen bestimmten Platz eines zweidimensionalen Feldes ansprechen will, dann müssen zusätzlich zum Variablennamen zwei Indizes (in Klammern hinter dem Variablennamen, durch ein Komma getrennt) angegeben werden:

- der erste Index für . . . . ,

- der zweite Index für . . . .

(3) Das Element "in der oberen, linken Ecke" der Darstellung auf Seite 23-09 soll ausgegeben werden. Wie lautet der Befehl?

100 . . . .

(4) Das Element "in der unteren, rechten Ecke" der Darstellung auf Seite 23-09 soll ausgegeben werden. Wie lautet der Befehl?

150 . . . .

(5) Ein Programmlauf wurde mit der folgenden Fehlermeldung unterbrochen:

```
Illegal data error in 150
Ready
```

An der fehlerverursachenden Stelle stehen die folgenden Befehle:

```
 .
 .
 .
120 DIM A(4, 5)
130 PRINT "ZUSCHLAG FUER"
140 PRINT "ABT. 6 / PLATZ 7";
150 INPUT A(6, 7)
 .
 .
 .
```

Versuchen Sie, die Fehlerursache zu erläutern!

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

LÖS 23.2

(1) Durch den Befehl

```
10 DIM T(5, 6)
```

werden (6 x 7 =) 42 Elemente reserviert.

(2) Wenn man einen bestimmten Platz eines zweidimensionalen Feldes ansprechen will, dann müssen zusätzlich zum Variablennamen zwei Indizes (in Klammern hinter dem Variablennamen, durch ein Komma getrennt) angegeben werden:

- der erste Index für die Zeilennummer,
- der zweite Index für die Spaltennummer.

(3) Das Element "in der oberen, linken Ecke" der Darstellung auf Seite 23-09 soll ausgegeben werden. Der Befehl lautet

```
100 PRINT A(0, 0)
```

(4) Das Element "in der unteren, rechten Ecke" der Darstellung auf Seite 23-09 soll ausgegeben werden. Der Befehl lautet

```
150 PRINT A(3, 4)
```

(5) Fehlerursache: Gem. der DIM-Anweisung in Zeile 120 reicht die Matrix A von A(0, 0) bis A(4, 5). Demnach wurde das in Zeile 150 angesprochene Element A(6, 7) nicht reserviert.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 23.1** !

ÜBUNGsprogramm zu Lektion 23

Programmbeschreibung (Wochentagsbestimmung):

Das Programm ermittelt zu einem einzugebenden Datum den Wochentag. Es ist jedoch zu beachten, daß das Datum nicht vor dem Jahre 1752 (Einführung des Gregorianischen Kalenders in England) liegen darf, da der Julianische Kalender, an dem man sich in den einzelnen Ländern unterschiedlich lange orientierte, anderen Regeln unterliegt als der heute noch gültige Gregorianische Kalender.

Programmliste:

```
10 REM *****************************
20 REM * BESTIMMUNG DES WOCHENTAGS *
30 REM *****************************
40 REM
50 REM ERLAEUTERUNG DES PROGRAMMZWECKS
60 REM ===============================
70 CLS
80 PRINT : PRINT : PRINT
90 PRINT " DIESES PROGRAMM ERMITTELT"
100 PRINT
110 PRINT " DEN WOCHENTAG"
120 PRINT
130 PRINT " ZU EINEM EINZUGEBENDEN"
140 PRINT
150 PRINT " DATUM."
160 FOR I = 1 TO 11 : PRINT : NEXT I
170 PRINT TAB(20) "'W'-TASTE DRUECKEN!"
180 GET A$ : IF A$ <> "W" THEN 180
190 REM
```

```
200 REM ZUORDNUNG DER WOCHENTAGE
210 REM ========================
220 DIM WT$(7)
230 FOR I = 1 TO 7
240 READ WT$(I)
250 NEXT I
260 REM
270 REM EINGABE DES DATUMS
280 REM ==================
290 CLS
300 PRINT : PRINT : PRINT
310 PRINT TAB(6) "BITTE JEWEILS NUR EINE
 ZAHL"
320 PRINT
330 PRINT TAB(6) "ZAHL EINGEBEN!"
340 PRINT : PRINT : PRINT
350 PRINT TAB(6) "TAG    "
360 PRINT : PRINT : PRINT
370 PRINT TAB(6) "MONAT "
380 PRINT : PRINT : PRINT
390 PRINT TAB(6) "JAHR   "
400 REM EINGABE DES TAGS
410 REM ----------------
420 CURSOR 13, 9
430 INPUT T$
440 LET T = VAL(T$)
450 REM UEBERPRUEFUNG DES EINGEGEBENEN
460 REM TAGS
470 REM ------------------------------
480 IF (T < 1) + (T > 31) + (T <> INT(T)
) THEN 500
490 GOTO 580
500 CURSOR 13, 9
510 PRINT [2, 0] "EINGABE WIEDERHOLEN!"
520 FOR I = 1 TO 1500 : NEXT I
530 CURSOR 13, 9
540 PRINT "
550 GOTO 420
560 REM EINGABE DES MONATS
570 REM ------------------
580 CURSOR 13, 13
590 INPUT M$
600 LET M = VAL(M$)
```

```
610 REM UEBERPRUEFUNG DES EINGEGEBENEN
620 REM MONATS
630 REM ------------------------------
640 IF (M < 1) + (M > 12) + (M <> INT(M)
) THEN 660
650 GOTO 720
660 CURSOR 13, 13
670 PRINT [2, 0] "EINGABE WIEDERHOLEN!"
680 FOR I = 1 TO 1500 : NEXT I
690 CURSOR 13, 13
700 PRINT "
710 GOTO 580
720 REM EINGABE DES JAHRS
730 REM ----------------
740 CURSOR 13, 17
750 INPUT J$
760 LET J = VAL(J$)
770 REM UEBERPRUEFUNG DES EINGEGEBENEN
780 REM JAHRS
790 REM ------------------------------
800 IF J < 1752 THEN 820
810 GOTO 890
820 CURSOR 13, 17
830 PRINT [2, 0] "ANGABE MUSS > 1751 SEI
N!"
840 FOR I = 1 TO 1500 : NEXT I
850 CURSOR 13, 17
860 PRINT "
870 GOTO 740
880 REM
890 REM BERECHNUNG DES WOCHENTAGS
900 REM =========================
910 LET K = INT(.6 + 1 / M)
920 LET L = J - K
930 LET A = M + 12 * K
940 LET P = L / 100
950 LET Z1 = INT(P / 4)
960 LET Z2 = INT(P)
970 LET Z3 = INT(5 * L / 4)
980 LET Z4 = INT(13 * (A + 1) / 5)
990 LET Z = Z4 + Z3 - Z2 + Z1 + T - 1
1000 LET Z  = Z - 7 * INT(Z / 7) + 1
1010 REM
```

```
1020 REM AUSGABE DES ERMITTELTEN
1030 REM WOCHENTAGS
1040 REM ======================
1050 CLS
1060 PRINT : PRINT : PRINT
1070 PRINT TAB(6) "DER  "; T; ".  "; M;
".  "; J
1080 PRINT : PRINT
1090 PRINT TAB(6) "WAR (IST) EIN"
1100 PRINT : PRINT
1110 PRINT [2, 0] TAB(10) WT$(Z); "."
1120 FOR I = 1 TO 10 : PRINT : NEXT I
1130 PRINT TAB(6) "WEITERE BERECHNUNGEN
(J/N)"
1140 GET A$
1150 IF A$ = "J" THEN 270
1160 IF A$ = "N" THEN 1180
1170 GOTO 1140
1180 END
1190 DATA SONNTAG, MONTAG, DIENSTAG
1200 DATA MITTWOCH, DONNERSTAG
1210 DATA FREITAG, SONNABEND
```

24. Vordefinierte Funktionen

Kaufmännisches Runden mit INT(X)

LE 24.1

Der Begriff "Funktion" wird in der BASIC-Literatur uneinheitlich und nicht unbedingt nur im mathematischen Sinne verwendet.

Die BASIC-Sprache enthält zwei Arten von Funktionen:

(1) Vordefinierte Funktionen,

(2) Selbstdefinierte (= vereinbarte) Funktionen.

Wir wollen uns zuerst den vordefinierten Funktionen zuwenden.

Was sind eigentlich vordefinierte Funktionen ?

Vordefinierte Funktionen sind reservierte BASIC-Wörter, durch die man im Rahmen eines umfassenden Befehls kleine, vorprogrammierte Maschinenprogramme für bestimmte Detailaufgaben aktiviert.

Beispiel:

```
 .
 .
300 LET C = SQR(B + A)
 .
 .
```

Der umfassende Befehl des Beispiels ist ".. LET C = ...", während der Funktionsausdruck durch "SQR(B + A)" dargestellt wird.

PE 24.1

(1) Welche zwei Arten von Funktionen enthält die BASIC-Sprache?

. . . .

. . . .

(2) Was sind v o r d e f i n i e r t e F u n k t i o n e n ?

. . . .

. . . .

. . . .

. . . .

. . . .

LÖS 24.1

(1) Die BASIC-Sprache enthält zwei Arten von Funktionen:

- Vordefinierte Funktionen,
- Selbstdefinierte Funktionen.

(2) Vordefinierte Funktionen sind reservierte BASIC-Wörter, durch die man im Rahmen eines umfassenden Befehls kleine, vorprogrammierte Maschinenprogramme für bestimmte Detailaufgaben aktiviert.

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 24.1** !

LE 24.2

In den vorangegangenen Lektionen haben wir schon einige vordefinierte Funktionen - wie z. B. SQR(X) oder CHR$(X) - verwendet. Zwar wird der einzelne Programmierer weitere Funktionen nutzen wollen, jedoch kaum jemals alle. Dazu sind die Anwendungsgebiete zu verschieden.

Aus diesem Grund wollen wir h i e r nur b e i s p i e l h a f t die Funktion

INT(X) (Abkürzung für engl. "integer" = ganze Zahl)

ausführlich erläutern und die anderen auf Seite 24-11 ff. in alphabetischer Reihenfolge kurz darstellen.

Mit Hilfe der Funktion INT(X) ermittelt man den Wert der größten ganzen Zahl, die den Wert von X nicht überschreitet. (Bei negativen Werten wird - wenn keine Ganzzahl vorliegt - die nächste kleinere Ganzzahl zur Verfügung gestellt.)

B e i s p i e l e (zur Wirkungsweise von INT(X):

```
10 PRINT INT(7.03)
20 PRINT INT(7.96)
30 PRINT INT(-8.03)
40 PRINT INT(-8.96)
RUN
 7
 7
-9
-9
Ready
```

Wozu benötigt man die Funktion INT(X) ?

Man benötigt sie am häufigsten im Rahmen eines umfassenderen Befehls, um Geldbeträge "kaufmännisch" zu runden. Wenn beispielsweise die Variable A den Wert 17.9658 enthält und der entsprechende, "kaufmännisch" gerundete Wert (also 17.97) in der Variablen B abgelegt werden soll, dann lautet hierzu der (umfassendere) BASIC-Befehl:

```
.. LET B = INT(A * 100 + 0.5) / 100
```

Wirkungsweise des Befehls:

1. Beispiel:

```
A (vor dem Runden)           =        17.9658

A * 100              ——————————►    1796.58
         + 0.5       ——————————►       0.5
                                    ———————
                                    1797.08

INT(A * 100 + 0.5)      ———————►    1797

INT(A * 100 + 0.5) / 100  ————►       17.97 ————► B
                                          (= kaufmännisch gerun-
                                          dete Zahl)
```

2. Beispiel:

```
A (vor dem Runden)         =        17.9648

A * 100          ──────────►      1796.48
      + 0.5      ──────────►         0.5
                                  ───────
                                  1796.98

INT(A * 100 + 0.5)    ──────►     1796

INT(A * 100 + 0.5) / 100 ───►      17.96 ─────► B
```

(= kaufmännisch gerundete Zahl)

PE 24.2

(1) Der Inhalt der Variablen X ist "kaufmännisch" zu runden und das Ergebnis in der Variablen E abzulegen! Wie lautet der entsprechende Befehl ?

100 ....

(2) Versuchen Sie herauszufinden, wie der Befehl lauten müßte, mit dem man den Inhalt der Variablen X auf drei Dezimalstellen genau auf- bzw. abrundet!

140 ....

(3) Versuchen Sie herauszufinden, wie der Befehl lauten müßte, mit dem man den Inhalt der Variablen X auf vier Dezimalstellen genau auf- bzw. abrundet!

180 ....

LÖS 24.2

(1) Der Inhalt der Variablen X ist "kaufmännisch" zu runden und das Ergebnis in der Variablen E abzulegen! Der entsprechende Befehl lautet

```
100 LET E = INT(X * 100 + 0.5) / 100   .
```

(2) Der Inhalt der Variablen X wird mit folgendem Befehl auf drei Dezimalstellen genau auf- bzw. abgerundet:

```
140 LET X = INT(X * 1000 + 0.5) / 1000   .
```

(3) Der Inhalt der Variablen X wird mit folgendem Befehl auf vier Dezimalstellen genau auf- bzw. abgerundet:

```
180 LET X = INT(X * 10000 + 0.5) / 10000   .
```

Sollten Ihre Lösungen nicht richtig sein, empfehlen wir Ihnen, in
→ **LE 24.2**
vor allem noch einmal die zwei Beispiele zur Wirkungsweise des Rundungsbefehls zu studieren!

LE 24.3

Die zahlreichen vordefinierten Funktionen, die beim SHARP MZ-700 zur Verfügung stehen, wollen wir nun in alphabetischer Reihenfolge kurz erläutern. Zwar sollten Sie sich merken, welche Funktionen es gibt, nicht aber ihre genaue Wirkungsweise. Darüber können Sie sich im Bedarfsfalle in dieser **LE 24.3** informieren!

Hinweis:

Wir verwenden die folgenden

Funktionsargumente	mit dieser Bedeutung:
X und Y	beliebige numerische Ausdrücke
I und J	Ganzzahlen
X$	String

Funktionsargumente müssen immer in Klammern eingeschlossen werden!

Funktion	**Erläuterungen**

ABS(X)

ermittelt den Absolutwert des Ausdrucks X. Der Absolutwert einer Zahl ist der Wert ohne ein Vorzeichen, also ohne "+" oder "-".

Beispiel:

```
10 LET X = 16742
20 PRINT ABS(-X)
30 PRINT ABS(+16)
40 PRINT ABS(-7)
RUN
 16742
 16
 7
Ready
```

ASC(X$)

ermittelt den ASCII-Dezimalwert des ersten Zeichens des Strings X$.

Ergänzender Hinweis:

Jedem Zeichen, also z. B. dem Buchstaben A oder der Ziffer 7 oder dem Sonderzeichen ! , entspricht im ASCII-Code ein bestimmtes Bit-Muster. Dem Dualwert dieses Bit-Musters entspricht wiederum ein bestimmter Dezimalwert. Dieser Dezimalwert, der gem. ASCII-Code-Tabelle einem bestimmten Zeichen zugeordnet ist, kann mit ASC(X$) ermittelt werden.

Beispiel:

```
10 LET A$ = "AHN CHA"
20 PRINT ASC(A$)
30 PRINT ASC("A")
40 PRINT ASC("7")
50 PRINT ASC("!")
RUN
 65
 65
 55
 33
Ready
```

Funktion	**Erläuterungen**

ATN(X)

ermittelt den Arcustangens von X im Bogenmaß (nicht in Graden!). Das Ergebnis liegt zwischen -pi/2 und +pi/2.

Beispiel:

```
10 INPUT X
20 PRINT ATN(X)
RUN
? 5
 1.3734008
Ready
```

CHR$(I)

ist gewissermaßen das "Gegenstück" zur Funktion ASC(X$). Mit der Funktion CHR$(I) ermittelt man das jeweilige Zeichen, das dem in Klammern angegebenen ASCII-Dezimalwert entspricht. I darf ein Wert von 0 - 255 sein; allerdings ist der Bereich von 0 - 32 nicht mit abdruckbaren Zeichen belegt.

Beispiel:

```
10 PRINT CHR$(65)
20 PRINT CHR$(55)
30 PRINT CHR$(33)
RUN
A
7
!
Ready
```

COS(X)

ermittelt den Kosinus des Argumentes X. X muß im Bogenmaß (nicht in Graden!) angegeben werden.

Funktion	Erläuterungen

Beispiel:

```
10 LET X = 45
20 LET A = COS(X * 0.0174533)
30 PRINT A
RUN
 .70710654
Ready
```

ERL — ist keine vordefinierte Funktion, sondern eine System-Variable. (Vgl. hierzu die Ausführungen auf Seite 22-38 f.)

ERN — ist keine vordefinierte Funktion, sondern eine System-Variable. (Vgl. hierzu die Ausführungen auf Seite 22-38 f.)

EXP(X) — berechnet die Exponentialfunktion von X zur Basis e (d. h. e^X), wobei e = 2.718281828... . X darf den Wert 88.029691 nicht überschreiten.

Beispiel:

```
10 PRINT EXP(5)
RUN
 148.41316
```

INT(X) — ermittelt den Wert der größten ganzen Zahl, die den Wert von X nicht überschreitet. Bei negativen Werten wird - wenn keine Ganzzahl vorliegt - die nächstkleinere Ganzzahl zur Verfügung gestellt. (Vgl. hierzu auch die Ausführungen auf Seite 24-05 ff.)

Funktion	Erläuterungen

Beispiel:

```
10 PRINT INT(7.03)
20 PRINT INT(7.96)
30 PRINT INT(-8.03)
40 PRINT INT(-8.96)
RUN
 7
 7
-9
-9
Ready
```

LEFT$(X$, I) — bewirkt, daß die mit I (0 - 255) angegebene Anzahl von Zeichen, von links beginnend, vom String X$ abgetrennt wird.

Beispiel:

```
10 LET A$ = "LOGIK DER PROGRAMMIERUNG"
20 PRINT LEFT$(A$, 5)
RUN
LOGIK
Ready
```

LEN(X$) — ermittelt die Anzahl der Zeichen eines Strings. Bei der Berechnung der Länge werden auch Leerzeichen (= Blanks) und nicht abdruckbare Zeichen mitgezählt.

Beispiel:

```
10 LET A$ = "EIN BUCH VON HAMANN"
20 PRINT LEN(A$)
RUN
 19
Ready
```

Funktion	**Erläuterungen**

LN(X)

ermittelt den natürlichen Logarithmus von X. X muß > 0 sein!
Achtung: Bei dieser - und der folgenden - Funktion gibt es erhebliche Abweichungen zwischen den verschiedenen BASIC-Versionen. Bei nicht wenigen Systemen berechnet man den natürlichen Logarithmus mit LOG(X) oder mit LOGE(X).

Beispiel:

```
10 INPUT "POSITIVE ZAHL EINGEBEN! "; P
20 LET L = LN(P)
30 PRINT "NAT. LOG. VON"; P; " IST"; L
RUN
POSITIVE ZAHL EINGEBEN! 100
NAT. LOG. VON 100 IST 4.6051702
Ready
```

LOG(X)

ermittelt den gewöhnlichen Logarithmus (Basis 10) von X für Werte X > 0.
Achtung: Bei dieser - und der vorangegangenen Funktion - gibt es erhebliche Abweichungen zwischen den verschiedenen BASIC-Versionen. Bei nicht wenigen Systemen berechnet man den gewöhnlichen Logarithmus mit LOG10(X) oder mit LGT(X); auch CLG(X) oder CLOG(X) sind gebräuchliche Formen für diese Funktion. (Vgl. ergänzend die Ausführungen zu LN(X)!)

Beispiel:

```
10 INPUT "POSITIVE ZAHL EINGEBEN! "; P
20 LET GL = LOG(P)
30 PRINT "GEW. LOG. VON"; P; " ="; GL
RUN
POSITIVE ZAHL EINGEBEN! 100
GEW. LOG. VON 100 = 2
Ready
```

Funktion	**Erläuterungen**
MID$(X$, I, J)	bewirkt, daß von einem bestimmten Zeichen ab, das mit I (0 - 255) gekennzeichnet ist, die mit J (0 - 255) angegebene Anzahl von Zeichen abgetrennt wird. Bei den Berechnungen werden auch Leerzeichen (= Blanks) und nicht abdruckbare Zeichen mitgezählt.

Beispiel:

```
10 LET A$ = "MAJA AHN CHA RIM HAMANN"
20 PRINT MID$(A$, 6, 11)
RUN
AHN CHA RIM
Ready
```

PAI(X)	multipliziert den Wert von X mit pi (= 3.14..). PAI(X) ist eine vordefinierte Funktion, die bei anderen BASIC-Versionen kaum jemals vorkommt.

Beispiel:

```
10 INPUT "ZAHL EINGEBEN! "; X
20 PRINT X; " * PI ="; PAI(X)
RUN
ZAHL EINGEBEN! 7.6
 7.6 * PI = 23.876104
Ready
```

PEEK(I) PEEK@(I)	ermittelt den Inhalt der Speicherstelle I (0 - 65535 dezimal oder $0000 - $FFFF hexadezimal) als Dezimalwert (0 - 255). Um an den Inhalt der Hauptspeicherstellen ab 53248 ($D000 hexadezimal) zu gelangen, ist - statt PEEK(I) - PEEK@(I) zu verwenden. Mit der Funktion PEEK(I) bzw. PEEK@(I) (und dem POKE-Befehl, s. u.) kann man - über das herkömmliche BASIC hinaus - Speicherplatzoperationen durchführen.

Funktion	**Erläuterungen**
	Anmerkung:
	Das "Gegenstück" zur Funktion PEEK(I) ist der Befehl POKE I, J. Er ermöglicht es, der Speicherstelle I (0 - 65535 dezimal oder $0000 - $FFFF hexadezimal) einen bestimmten Inhalt J (0 - 255) zuzuweisen. Zur Adressierung der Hauptspeicherstellen ab 53248 ($D000 hexadezimal) ist die Anweisung POKE@I, J zu verwenden.
	Der Anfänger sollte PEEK(I) / PEEK@(I) und POKE I, J / POKE@I, J nur im Ausnahmefall benutzen.
RAD(X)	wandelt X (in Grad) in Bogenmaß um. RAD(X) ist eine vordefinierte Funktion, die bei anderen BASIC-Versionen kaum jemals vorkommt. Allerdings gibt es bei einigen wenigen Systemen statt dessen die Anweisung RAD.

Beispiel:

```
10 INPUT "WERT IN GRAD? "; X
20 PRINT X; " GRAD ="; RAD(X); "BOGENMAS
S"
RUN
WERT IN GRAD? 30
 30 GRAD = .52359878 BOGENMASS
Ready
```

RIGHT$(X$, I)	bewirkt, daß die mit I (0 - 255) angegebene Anzahl von Zeichen, von rechts beginnend, vom String X$ abgetrennt wird.

Beispiel:

```
10 LET A$ = "INTERPRETATION ZU EFFI BRIE
ST"
20 LET B$ = "BUCH VON ELSBETH HAMANN"
30 PRINT RIGHT$(A$, 11)
40 PRINT RIGHT$(B$, 6)
```

Funktion	Erläuterungen

```
RUN
EFFI BRIEST
HAMANN
Ready
```

RND(X)

ermittelt eine Pseudo-Zufallszahl im Bereich von 0.00000001 bis 0.99999999.
Ist X > 0, liefert das System jeweils eine neue Zufallszahl. Ist X <= 0, wird der Zufallsgenerator zurückgesetzt (initialisiert) und der erste Wert der Pseudo-Zufallszahlen zur Verfügung gestellt. Diese Initialisierung ist immer dann wichtig, wenn man eine wiederholbare Folge von Zufallszahlen benötigt.

Beispiel:

```
10 FOR I = 1 TO 10
20 PRINT RND(I)
30 NEXT I
RUN
 .39065826
 .31561637
 .62537825
 .88934362
 .14231551
 .82160568
 .66099
 .92203057
 .35371029
 .91818686
Ready
```

Mit Hilfe der RND(X)-Funktion ist es beispielsweise möglich, einen Würfel zu simulieren. Da der Würfel sechs verschiedene Werte hat, multiplizieren wir die durch RND(1) generierte Zufallszahl mit sechs: (RND(1) * 6), was eine Zahl zwischen 0.00.. und 5.9.. ergibt. Wir erhöhen deshalb das Produkt um 1: (RND(1) * 6 + 1). Mit der INT(X)-Funktion werden dann die "Nach-dem-Komma-Stellen" abgetrennt: INT(RND(1) * 6 + 1).

Funktion **Erläuterungen**

Beispiel:

```
10 PRINT "10 WUERFELERGEBNISSE:"
20 FOR I = 1 TO 10
30 PRINT INT(RND(1) * 6 + 1);
40 NEXT I
RUN
 4 2 1 6 5 1 6 6 3 4
Ready
```

SGN(X)

erzeugt den Wert 1, wenn X > 0;
erzeugt den Wert 0, wenn X = 0;
erzeugt den Wert -1, wenn X < 0.

Beispiel:

```
100 ON SGN(EK)+2 GOSUB 1000, 2000, 3000
```

(Mit diesem Befehl verzweigt man in drei verschiedene Unterprogramme, und zwar, wenn EK (= End Kapital) negativ ist, nach Zeile 1000, wenn EK = 0 ist, nach 2000, wenn EK positiv ist, nach 3000.)

SIN(X)

ermittelt den Sinus des Argumentes X. X muß im Bogenmaß (nicht in Graden!) angegeben werden.

Beispiel:

```
10 INPUT A
20 PRINT SIN(A)
RUN
? 2
 .90929743
Ready
```

SIZE

ist keine vordefinierte Funktion, sondern eine System-Variable. (Vgl. hierzu die Ausführungen auf Seite 8-17!)

Funktion **Erläuterungen**

SPC(I)

erzeugt I (0 - 255) Leerstellen beim Ausgabegerät. SPC(I) darf nur mit P R I N T verwendet werden.

Beispiel:

```
10 REM UNTERSCHIED ZWISCHEN
20 REM -------------------
30 REM TAB(I) UND SPC(I)
40 REM -----------------
50 PRINT "          11111111112"
60 PRINT "012345678901234567890"
70 PRINT TAB(5) "A"; TAB(10) "B"
80 PRINT SPC(5) "A"; SPC(10) "B"
RUN
          11111111112
012345678901234567890
     A    B
     A          B
Ready
```

SQR(X)

ermittelt die Quadratwurzel von X. X muß >= 0 sein.

Beispiel:

```
10 FOR I = 0 TO 10
20 PRINT "QUAD. W. V."; I; " ="; SQR(I)
30 NEXT I
RUN
QUAD. W. V. 0 = 0
QUAD. W. V. 1 = 1
QUAD. W. V. 2 = 1.4142136
QUAD. W. V. 3 = 1.7320508
QUAD. W. V. 4 = 2
QUAD. W. V. 5 = 2.236068
QUAD. W. V. 6 = 2.4494897
QUAD. W. V. 7 = 2.6457513
QUAD. W. V. 8 = 2.8284271
QUAD. W. V. 9 = 3
QUAD. W. V. 10 = 3.1622777
```

Funktion	Erläuterungen

STR$(X)

wandelt eine Zahl oder eine numerische Variable in einen String um.

Beispiel:

```
10 INPUT "ZAHL EINGEBEN "; X
20 LET B$ = STR$(X)
30 PRINT "DIE ZAHL BESTEHT AUS";
40 PRINT LEN(B$); " ZEICHEN."
RUN
ZAHL EINGEBEN 265.34
DIE ZAHL BESTEHT AUS 6 ZEICHEN.
Ready
```

Hinweis:

Bei der Umwandlung einer Zahl in einen String mit Hilfe von STR$(X) wird zwar ein führendes Minuszeichen berücksichtigt, nicht hingegen die führende Leerstelle eines positiven Wertes. Diesbezüglich bestehen aber Abweichungen zwischen den verschiedenen BASIC-Versionen.

TAB(I)

bewirkt einen Tabulator-Sprung an die Druckstelle I (0 - 255). Es ist zu beachten, daß die erste Druckposition 0 (und nicht etwa 1) ist! TAB(I) darf nur mit PRINT oder PRINT/P verwendet werden.

Beispiel:

```
10 PRINT "01234567890"
20 PRINT TAB(0) "A";
30 PRINT TAB(5) "B";
40 PRINT TAB(9) "C";
RUN
01234567890
A    B   C
Ready
```

Funktion	**Erläuterungen**
TAN(X)	ermittelt den Tangens des Argumentes X. X muß im Bogenmaß (nicht in Graden!) angegeben werden.

Beispiel:

```
10 INPUT A
20 PRINT TAN(A)
RUN
? 0.5
.54630249
Ready
```

TI$	ist keine vordefinierte Funktion, sondern eine System-Variable. (Vgl. hierzu die Ausführungen auf Seite 8-17!)
VAL(X$)	Mit Strings kann man nicht rechnen, auch wenn sie Ziffern enthalten. Die Funktion VAL(X$) ermöglicht es, führende Ziffern eines Strings in eine Zahl umzuformen. (Den Ziffern darf neben führenden Leerstellen noch das Zeichen + und/oder das Zeichen - und/oder das Zeichen . vorangestellt sein.) Sind die führenden Zeichen keine Ziffern, dann ist VAL(X$) = 0.

Beispiel:

```
10 LET A$ = "7 ZWERGE"
20 LET B$ = "5 RIESEN"
30 PRINT A$; " UND "; B$
40 PRINT "SIND ZUSAMMEN"
50 PRINT VAL(A$) + VAL(B$);
60 PRINT " FABELWESEN."
RUN
7 ZWERGE UND 5 RIESEN
SIND ZUSAMMEN
 12 FABELWESEN.
Ready
```

PE 24.3

Kennzeichnen Sie jene Wörter durch ein Kreuz, die eine vordefinierte BASIC-Funktion darstellen!

SHARP(A$)	
ABS(X)	
ASC(X$)	
KEIICHI(X)	
SHIMIZU(X$, I, J)	
ATN(X)	
HAMANN(X$)	
CHR$(I)	
COS(X)	
OSKAR(X$)	
EICHLER(X$, I)	
ERL	
ERN	
EXP(X)	
MASAAKI(COS)	
INT(X)	
KANEKO(X$)	
LEFT$(X$, I)	
LEN(X$)	

FESSEL(X$)	
MAJA(X)	
AHN(X$, I)	
CHA(X)	
RIM(X$)	
LN(X)	
LOG(X)	
COBOL(X$)	
MID$(X$, I, J)	
PEEK(I)	
POKE(I, J)	
PAI(X)	
RAD(X)	
RIGHT$(X$, I)	
RND(X)	
ELSBETH(X$)	
SGN(X)	
SIN(X)	
SIZE	
SPC(I)	
SQR(X)	
STR$(X)	
TAB(I)	
TAN(X)	
VAL(X$)	

LÖS 24.3

Die durch ein Kreuz gekennzeichneten Wörter sind vordefinierte BASIC-Funktionen!

SHARP(A$)	
ABS(X)	X
ASC(X$)	X
KEIICHI(X)	
SHIMIZU(X$, I, J)	
ATN(X)	X
HAMANN(X$)	
CHR$(I)	X
COS(X)	X
OSKAR(X$)	
EICHLER(X$, I)	
ERL	
ERN	
EXP(X)	X
MASAAKI(COS)	
INT(X)	X
KANEKO(X$)	
LEFT$(X$, I)	X
LEN(X$)	X

FESSEL(X$)	
MAJA(X)	
AHN(X$, I)	
CHA(X)	
RIM(X$)	
LN(X)	X
LOG(X)	X
COBOL(X$)	
MID$(X$, I, J)	X
PEEK(I)	X
POKE(I, J)	
PAI(X)	X
RAD(X)	X
RIGHT$(X$, I)	X
RND(X)	X
ELSBETH(X$)	
SGN(X)	X
SIN(X)	X
SIZE	
SPC(I)	X
SQR(X)	X
STR$(X)	X
TAB(I)	X
TAN(X)	X
VAL(X$)	X

Sollten Ihre Lösungen nicht richtig sein, ist es trotzdem nicht unbedingt erforderlich, die **LE 24.3** nochmals durchzuarbeiten. Informieren Sie sich besser im Bedarfsfall!

ÜBUNGsprogramm zu Lektion 24

Programmbeschreibung (Training des Kopfrechnens):

Das Programm ist dafür vorgesehen, mit Kindern Kopfrechnen - wahlweise Addition, Subtraktion, Multiplikation oder Division - zu üben. Der Anwender muß sich für eine der Rechenarten entscheiden und anschließend den Höchstwert für die 1. und dann für die 2. Zahl bestimmen.

Bei den Subtraktionsaufgaben haben wir durch den Befehl der Zeile 3050 sichergestellt, daß die 2. Zahl niemals größer ist als die 1. Zahl. (Wenn diese Nebenbedingung unerwünscht ist, braucht man nur den Befehl der Zeile 3050 zu löschen!)

Bei den Divisionsaufgaben haben wir zusätzlich folgendes berücksichtigt:

(1) Für den Dividenden (= 1. Zahl) und für den Divisor (= 2. Zahl) ist der Wert 0 ausgeschlossen (Zeilen 5030 und 5040).

(2) Der Divisor ist nie größer als der Dividend (Zeile 5050).

(3) Es sind nur Aufgaben mit ganzzahligem Ergebnis vorgesehen (Zeile 5060).

Programmliste:

```
10 REM ******************************
20 REM * TRAINING DES KOPFRECHNENS *
30 REM ******************************
40 REM
50 REM DIESES PROGRAMM IST MEINER
60 REM TOCHTER MAJA GEWIDMET.
70 REM
```

```
80 REM BESTIMMUNG DER RECHENART
90 REM ========================
100 CLS
110 PRINT : PRINT : PRINT
120 PRINT " WAS WOLLEN WIR UEBEN?"
130 PRINT : PRINT : PRINT
140 PRINT " ADDITION          (1)"
150 PRINT : PRINT
160 PRINT " SUBTRAKTION       (2)"
170 PRINT : PRINT
180 PRINT " MULTIPLIKATION    (3)"
190 PRINT : PRINT
200 PRINT " DIVISION          (4)"
210 PRINT : PRINT : PRINT
220 PRINT TAB(10) "NUR EINE ZIFFER, NAEM
LICH"
230 PRINT
240 PRINT TAB(10) "1, 2, 3 ODER 4 ANGEBE
N!"
250 GET A$
260 IF (A$<>"1") * (A$<>"2") * (A$<>"3")
 * (A$<>"4") THEN 250
270 REM
280 REM HOECHSTWERTBESTIMMUNG
290 REM ====================
300 CLS
310 FOR I = 1 TO 10 : PRINT : NEXT I
320 PRINT " HOECHSTWERT FUER 1. ZAHL ";
330 INPUT Z1
340 PRINT : PRINT : PRINT
350 PRINT " HOECHSTWERT FUER 2. ZAHL ";
360 INPUT Z2
370 REM
380 REM VERZWEIGUNG ZUR GEWUENSCHTEN
390 REM RECHENART
400 REM ===========================
410 REM
420 ON VAL(A$) GOTO 2000,3000,4000,5000
430 REM
2000 REM ADDITIONSUEBUNGEN
2010 REM =================
2020 REM
2030 LET X = INT(RND(1) * (Z1 + 1))
2040 LET Y = INT(RND(1) * (Z2 + 1))
```

```
2050 CLS
2060 FOR I = 1 TO 10 : PRINT : NEXT I
2070 PRINT " WIEVIEL IST"
2080 PRINT : PRINT
2090 PRINT X; " +"; Y; " = ";
2100 INPUT ""; E
2110 IF E = (X+Y) THEN GOTO 2160
2120 REM FALSCHES ERGEBNIS:
2130 REM -----------------
2140 GOSUB 10000 : REM UP-FALSCH-MELDUNG
2150 GOTO 2070
2160 REM RICHTIGES ERGEBNIS:
2170 REM ------------------
2180 GOSUB 20000 : REM UP-RICHT.-MELDUNG
2190 IF WH$ = "J" THEN 2000
2200 IF WH$ = "N" THEN 6000
2210 REM
3000 REM SUBTRAKTIONSUEBUNGEN
3010 REM ====================
3020 REM
3030 LET X = INT(RND(1) * (Z1 + 1))
3040 LET Y = INT(RND(1) * (Z1 + 1))
3050 IF Y > X THEN 3040
3060 CLS
3070 FOR I = 1 TO 10 : PRINT : NEXT I
3080 PRINT " WIEVIEL IST"
3090 PRINT : PRINT
3100 PRINT X; " -"; Y; " = ";
3110 INPUT ""; E
3120 IF E = (X-Y) THEN GOTO 3170
3130 REM FALSCHES ERGEBNIS:
3140 REM -----------------
3150 GOSUB 10000 : REM UP-FALSCH-MELD.
3160 GOTO 3080
3170 REM RICHTIGES ERGEBNIS:
3180 REM ------------------
3190 GOSUB 20000  : REM UP-RICHT.-MELD.
3200 IF WH$ = "J" THEN 3000
3210 IF WH$ = "N" THEN 6000
3220 REM
4000 REM MULTIPLIKATIONSUEBUNGEN
4010 REM =======================
4020 REM
4030 LET X = INT(RND(1) * (Z1 + 1))
4040 LET Y = INT(RND(1) * (Z2 + 1))
```

```
4050 CLS
4060 FOR I = 1 TO 10 : PRINT : NEXT I
4070 PRINT " WIEVIEL IST"
4080 PRINT : PRINT
4090 PRINT X; " *"; Y; " = ";
4100 INPUT ""; E
4110 IF E = (X*Y) THEN GOTO 4180
4120 REM FALSCHES ERGEBNIS:
4130 REM ------------------
4140 GOSUB 10000 : REM UP-FALSCH-MELD.
4150 GOTO 4070
4160 REM RICHTIGES ERGEBNIS:
4170 REM -------------------
4180 GOSUB 20000 : REM UP-RICHT.-MELD.
4190 IF WH$ = "J" THEN 4000
4200 IF WH$ = "N" THEN 6000
4210 REM
5000 REM DIVISIONSUEBUNGEN
5010 REM =================
5020 REM
5030 LET X = INT(RND(1) * Z1 + 1)
5040 LET Y = INT(RND(1) * Z2 + 1)
5050 IF Y > X THEN 5040
5060 IF X/Y > INT(X/Y) THEN 5030
5070 CLS
5080 FOR I = 1 TO 10 : PRINT : NEXT I
5090 PRINT " WIEVIEL IST"
5100 PRINT : PRINT
5110 PRINT X; " :"; Y; " = ";
5120 INPUT ""; E
5130 IF E = (X/Y) THEN 5200
5140 REM FALSCHES ERGEBNIS:
5150 REM ------------------
5160 GOSUB 10000 : REM UP-FALSCH-MELD.
5170 GOTO 5090
5180 REM RICHTIGES ERGEBNIS:
5190 REM -------------------
5200 GOSUB 20000 : REM UP-RICHTIG-MELD.
5210 IF WH$ = "J" THEN 5000
5220 IF WH$ = "N" THEN 6000
5230 REM
```

```
6000 REM ZUERUECK ZUM MENUE ?
6010 REM ====================
6020 CLS
6030 FOR I = 1 TO 10 : PRINT : NEXT I
6040 PRINT " ZURUECK ZUR WAHL DER AUFGAB
ENART ?"
6050 PRINT
6060 PRINT " (J/N)"
6070 GET A$
6080 IF A$ = "J" THEN 80
6090 IF A$ = "N" THEN 6110
6100 GOTO 6070
6110 END
10000 REM * UP-FALSCH-MELDUNG*
10010 REM ====================
10020 CLS
10030 PRINT : PRINT : PRINT
10040 PRINT " F A L S C H  !!!"
10050 PRINT : PRINT
10060 PRINT " DESHALB NOCH EINMAL:"
10070 PRINT : PRINT : PRINT : PRINT
10080 RETURN
10090 REM
20000 REM * UP-RICHT.-MELDUNG *
20010 REM ====================
20020 CLS
20030 FOR I = 1 TO 10 : PRINT : NEXT I
20040 PRINT " R  I  C  H  T  I  G  !"
20050 PRINT : PRINT
20060 PRINT " WEITERE AUFGABEN (J/N)?"
20070 GET WH$
20080 IF (WH$ = "J") + (WH$ = "N") THEN
GOTO 20100
20090 GOTO 20070
20100 RETURN
```

25. Selbstdefinierte Funktionen mit DEF FN

LE 25

In der vorangegangenen Lektion 24 haben wir zahlreiche vordefinierte Funktionen kennengelernt. Darüber hinaus ist es aber möglich, eigene Funktionen zu definieren.

Wozu benötigt man selbstdefinierte Funktionen ?

Wird z. B. in einem Programm ein Algorithmus mehrmals in der gleichen Form, aber mit unterschiedlichen Variablen benötigt, dann kann dieser vom Benutzer als Funktion definiert und anschließend an verschiedenen Programmstellen aufgerufen werden.

Beispiel:

```
 .
 .
 .
120 LET E1 = AK * (1 + Z1 * FJ / 100)
 .
 .
 .
270 LET E2 = S * (1 + Z2 * J / 100)
 .
 .
 .
430 LET E3 = GK * (1 + Z3 * A / 100)
 .
 .
 .
```

In dem aufgeführten Beispiel sind die Voraussetzungen für den sinnvollen Einsatz einer selbstdefinierten Funktion erfüllt, weil

- ein Rechenvorgang mehrmals in der gleichen Form, aber
- mit unterschiedlichen Variablen

durchzuführen ist.

Man definiert eine anwenderspezifische Funktion mit dem Befehl

DEF FN (Abkürzung für engl. "to define a function = eine Funktion definieren).

Für unser Beispiel könnte der Befehl zur Funktionsdefinition folgendermaßen lauten:

```
115 DEF FNA(X, Y, Z) = X * (1 + Y * Z / 100)
└─┘ └─┘ └─┘└───────┘   └──────────────────┘
(1) (2) (3)   (4)              (5)
```

Zu (1): Bevor man eine anwenderspezifische Funktion aufruft, muß man sie definiert haben, d. h. die in Zeile 115 definierte Funktion kann nur in einer Zeile > 115 benutzt werden.

Zu (2): DEF ist der Operationsteil des Befehls. Hierdurch weiß der Rechner, "was" er machen soll.

Zu (3): Der in unserem Beispiel definierte Funktionsname ist FNA. Er setzt sich zusammen aus einem beliebigen Variablennamen (hier: A), dem immer die Buchstaben FN vorangestellt sein müssen.

Zu (4): Dies ist die sog. Parameterliste. Sie besteht aus Variablennamen, die durch die aktuellen (jeweils unterschiedlichen) Variablen ersetzt werden, wenn die Funktion aufgerufen wird.

Zu (5): Dies ist die definierte Funktion.

Die von uns definierte Funktion

```
115 DEF FNA(X, Y, Z) = X * (1 + Y * Z / 100)
```

kann nun in folgenden Zeilen sehr einfach aufgerufen werden:

```
.
.
.
120 LET E1 = FNA(AK, Z1, FJ)
.
.
.
270 LET E2 = FNA(S, Z2, J)
.
.
.
430 LET E3 = FNA(GK, Z3, A)
.
.
.
```

Die Funktionsaufrufe ersetzen die im Beispiel aufgeführten Rechenbefehle. Man spart dadurch viel Schreibarbeit und vermindert die Fehlerwahrscheinlichkeit.

Ergänzende Hinweise:

(1) Die Variablen einer DEF FN-Anweisung sind nur für die Rechenvorschrift innerhalb einer DEF FN-Anweisung von Bedeutung, d. h. sie haben keinen Einfluß auf Variablen im Programm, die den gleichen Namen haben. Man spricht daher auch von "Scheinvariablen" oder "Formalparametern".

(2) Die definierte Funktion darf natürlich auch - wie andere mathematische Ausdrücke - Variablen enthalten, die nicht in der Parameterliste erscheinen.

PE 25

In einem Programm erscheinen die folgenden Rechenbefehle:

```
     .
     .
     .
180 LET Q1 = X1 ↑ X2 * Z1 / 100
     .
     .
     .
290 LET Q2 = A1 ↑ A2 * Z2 / 100
     .
     .
     .
340 LET Q3 = B1 ↑ B2 * Z3 / 100
     .
     .
     .
```

Da in allen drei Befehlen ein Rechenvorgang mehrmals in der gleichen Form, aber mit unterschiedlichen Variablen durchzuführen ist, sollen Sie in Zeile 10 eine anwenderspezifische Funktion definieren (Funktionsname: FNHA, Formalparameter: D1, D2, D3) und in den Zeilen 180, 290, 340 statt der obigen Rechenbefehle die selbstdefinierte Funktion aufrufen!

```
10 ......................................
 .
 .
 .
180 .....................................
 .
 .
 .
290 .....................................
 .
 .
 .
340 .....................................
 .
 .
 .
```

LÖS 25

Die Befehle haben folgendes Aussehen:

```
10 DEF FNHA(D1, D2, D3) = D1 ↑ D2 * D3 / 100
.
.
.
180 LET Q1 = FNHA(X1, X2, Z1)
.
.
.
290 LET Q2 = FNHA(A1, A2, Z2)
.
.
.
340 LET Q3 = FNHA(B1, B2, Z3)
.
.
.
```

Sollte Ihre Lösung nicht richtig sein, kehren Sie zurück zur

→ **LE 25** !

ÜBUNGsprogramm zu Lektion 25

Problemstellung:

Während man mit der vordefinierten Funktion CHR$(I) zu einem Dezimalwert von 0 - 255 das zugeordnete Zeichen gem. ASCII-Code ermittelt, ist es mit ASC(X$) möglich, zu eingegebenen Zeichen den entsprechenden Dezimalwert festzustellen.

Programmbeschreibung (Dezimalwert eines beliebigen Zeichens):

Das folgende Programm wurde so konzipiert, daß nicht nur zu den abdruckbaren Zeichen die ASCII-Dezimalzahl ausgegeben wird, sondern auch zu [CR], [DEL], [INST] etc.

Programmliste:

```
10 REM ***********************
20 REM *  DEZIMALWERT EINES   *
30 REM * BELIEBIGEN ZEICHENS *
40 REM ***********************
50 CLS
60 PRINT
70 PRINT " BITTE ZEICHEN/TASTE BETAETIGE
N!"
80 PRINT : PRINT
90 GET A$ : IF A$ = "" THEN 90
100 PRINT " DAS EINGEGEBENE ZEICHEN"
110 PRINT
120 PRINT " (DIE BETAETIGTE TASTE)"
130 PRINT
140 PRINT " ENTSPRICHT FOLG. ASCII-DEZIM
ALWERT:"
150 PRINT
```

```
160 PRINT TAB(10) ASC(A$)
170 FOR I = 1 TO 10 : PRINT : NEXT I
180 PRINT TAB(14) "WEITERE EINGABEN ? (J
/N)"
190 GET JN$
200 IF JN$ = "J" THEN 50
210 IF JN$ = "N" THEN 230
220 GOTO 190
230 END
```

26. Gestaltung der Ausgabe mit PRINT USING

LE 26.1

Schon in Lektion 12 hatten wir einige Alternativen zur Darbietung der Ausgabeergebnisse kennengelernt:

- PRINT (und PRINT/P) mit Formatierung durch Semikola (;) ,
- PRINT (und PRINT/P) mit Formatierung durch Kommata (,) ,
- PCOLOR ../ MODE TN / MODE TL / MODE TS i. V. m. PRINT/P.

Eine erhebliche Erweiterung der Gestaltungsmöglichkeiten ergab sich aus den zahlreichen in Lektion 16 ("Farbe und Grafik") vorgestellten Befehlen.

Außerdem sind insbesondere die vordefinierten Funktionen

- SPC(I) (vgl. Seite 24-22),
- TAB(I) (vgl. Seite 24-23),

aber (u. a.) auch

- ABS(X) (vgl. Seite 24-13),
- CHR$(I) (vgl. Seite 24-14 i. V. m. der Fußnote auf Seite 8-11 f.),
- INT(X) (vgl. Seite 24-15 i. V. m. Seite 24-05 ff.),
- LEFT$(X$, I) (vgl. Seite 24-16),
- LEN(X$) (vgl. Seite 24-16),
- MID$(X$, I, J) (vgl. Seite 24-18),

- RIGHT$(X$, I) (vgl. Seite 24-19 f.),
- STR$(X) (vgl. Seite 24-23),
- VAL(X$) (vgl. Seite 24-24)

für eine zweckmäßige Aufbereitung der Daten zu verwenden.

Darüber hinaus steht uns noch der Befehl

PRINT USING .. (von engl. "print using", hier zu übersetzen mit "gib aus unter Verwendung von ..")

zur Verfügung. Er gehört zu den mächtigsten und vielseitigsten Anweisungen, die es in BASIC gibt. Er dient der formatierten Datenausgabe unter Verwendung einer Druckmaske.[1]

Was versteht man unter einer "Druckmaske"?

Eine Druckmaske ist eine Folge von bestimmten Zeichen, mit deren Hilfe wir das Erscheinungsbild von Ausgabedaten festlegen.

Beispiel:

```
 .
 .
120 PRINT USING "####.##"; A
 .
 .
```

Bedeutung:

120 Gib aus unter Verwendung von "####.##" (den Inhalt von) A!
(Druckmaske: "####.##")

1) Bei Verwendung von PRINT USING erfolgt die Ausgabe auf dem angeschlossenen Bildschirm. Eine Datenausgabe auf dem angeschlossenen Plotter-Drucker führt man mit dem Befehl

PRINT/P USING .. (von engl. "print at the printer using", hier zu übersetzen mit "gib auf dem Drucker aus unter Verwendung von ..)

herbei. Die weiteren Ausführungen zu PRINT USING gelten für ihn in analoger Weise.

Auf die Bedeutung und Wirkung dieser (und anderer) Druckmaskenzeichen werden wir in **LE 26.2** und **LE 26.3** näher eingehen.

PE 26.1

Welche der folgenden Behauptungen ist/sind richtig?

A Eine Gestaltung von Ausgabeergebnissen ist nur mit der Anweisung PRINT USING möglich.

B Für die Gestaltung der Ausgabe stehen zahlreiche Befehle und vordefinierte Funktionen zur Verfügung.

C Der Befehl PRINT USING dient der formatierten Datenausgabe unter Verwendung einer Druckmaske.

D Eine Druckmaske ist eine Folge von bestimmten Zeichen, mit deren Hilfe wir das Erscheinungsbild von Ausgabedaten festlegen können.

E Eine Druckmaske ist eine Folge von bestimmten Zeichen, mit deren Hilfe geheime Daten zum Zwecke der Unkenntlichmachung überschrieben werden.

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

LÖS 26.1

Die Lösungsbuchstaben lauten

B, C, D .

Sollte Ihre Lösung nicht richtig sein, kehren Sie zurück zur
→ **LE 26.1** !

LE 26.2

Im Rahmen des PRINT USING-Befehls stehen zahlreiche - mehr oder weniger wichtige - Druckmaskenzeichen zur Verfügung. Wir wollen sie nacheinander immer zuerst anhand eines Beispiels vorstellen, um sie dann zu erläutern.

Beispiel 1 (Druckmaskenzeichen # und .):

```
10 FOR X = 1 TO 5
20 READ N
30 PRINT USING "####.##"; N
40 NEXT X
50 DATA .6, .678, 12.3, 1495.556, 9768
RUN
   0.60
   0.68
  12.30
1495.56
9768.00
Ready
```

Erläuterungen zu Beispiel 1:

(1) Anführungsstriche kennzeichnen Beginn und Ende der Druckmaske, die vom folgenden Operanden durch ein Semikolon zu trennen ist.

(2) Das Druckmaskenzeichen # beschreibt eine Dezimalzifferstelle, das Druckmaskenzeichen . die Position des Dezimalpunkts.

(3) Rechts vom Dezimalpunkt erscheinen immer so viele Ziffern, wie durch die #-Zeichen vorgegeben ist, und zwar
- ggf. korrekt auf- bzw. abgerundet,
- ggf. ergänzt mit nachgezogenen Nullen.

(4) Besteht die zu druckende Zahl (links vom Dezimalpunkt) aus weniger Ziffern als der Druckmaske entspricht, dann ergänzt das System eine entsprechende Anzahl von Leerstellen (= "rechtsbündige" Ausgabe).

(5) Enthält die Druckmaske links vom Dezimalpunkt mindestens ein #-Zeichen, dann erscheint - wenn die auszugebende Zahl nur aus Dezimalen besteht - links vom Dezimalpunkt eine 0 (Null).

Beispiel 2 (Druckmaskenzeichen # und . in Verbindung mit der Funktion TAB(I) und dem Formatierungszeichen ;):

```
10 LET A = 7
20 LET B = 188.5
30 LET C = 366.897
40 LET D = .76
50 PRINT "          1111111111222222"
60 PRINT "0123456789012345678901234 5"
70 PRINT USING "###.##"; A;
80 PRINT TAB(10) USING "###.##"; B
90 PRINT USING "###.##"; C;
100 PRINT TAB(10) USING "###.##"; D
RUN
          1111111111222222
01234567890123456789012345
  7.00    188.50
366.90      0.76
Ready
```

Erläuterungen zu Beispiel 2:

(1) Das letzte Zeichen in Zeile 70 und das gleiche in Zeile 90 ist das Formatierungszeichen ; . Es verhindert den Zeilenvorschub. (Vgl. Seite 12-07 !)

(2) Durch kombinierte Anwendung der Druckmaskenzeichen # und . , des Formatierungszeichens ; und der Funktion TAB(I) ist der Ausdruck beliebiger, übersichtlich gestalteter Tabellen möglich, bei denen die Zahlen dezimalpunktgerecht untereinanderstehen.

Beispiel 3 (Druckmaskenzeichen + als erstes Zeichen der Druckmaske):

```
10 PRINT "+ ALS ERSTES ZEICHEN DER DRUCK
MASKE"
20 FOR X = 1 TO 5
30 READ N
40 PRINT USING "+####.##"; N
50 NEXT X
60 DATA .6, -.678, -12.3
70 DATA 1495.556, -9768
RUN
+ ALS ERSTES ZEICHEN DER DRUCKMASKE
    +.60
    -.68
  -12.30
+1495.56
-9768.00
Ready
```

Erläuterungen zu Beispiel 3:

Das Zeichen + zu Beginn der Druckmaske bewirkt, daß den ausgegebenen Zahlen grundsätzlich das korrekte Vorzeichen (+ oder -) vorangestellt ist.

Beispiel 4 (Druckmaskenzeichen + als letztes Zeichen der Druckmaske):

```
10 PRINT "+ ALS LETZTES ZEICHEN DER DRUC
KMASKE"
20 FOR X = 1 TO 5
30 READ N
40 PRINT USING "####.##+"; N
50 NEXT X
60 DATA .6, -.678, -12.3
70 DATA 1495.556, -9768
```

```
RUN
+ ALS LETZTES ZEICHEN DER DRUCKMASKE
   0.60+
   0.68-
  12.30-
1495.56+
9768.00-
Ready
```

Erläuterungen zu Beispiel 4:

Das Zeichen + am Ende der Druckmaske bewirkt, daß den ausgegebenen Zahlen grundsätzlich das korrekte "Vorzeichen" (+ oder -) nachgestellt ist.

Beispiel 5 (Druckmaskenzeichen - als letztes Zeichen der Druckmaske):

```
10 PRINT "- ALS LETZTES ZEICHEN DER DRUC
KMASKE"
20 FOR X = 1 TO 5
30 READ N
40 PRINT USING "####.##-"; N
50 NEXT X
60 DATA .6, -.678, -12.3
70 DATA 1495.556, -9768
RUN
- ALS LETZTES ZEICHEN DER DRUCKMASKE
   0.60
   0.68-
  12.30-
1495.56
9768.00-
Ready
```

Erläuterungen zu Beispiel 5:

Das Zeichen - am Ende der Druckmaske bewirkt, daß (nur) den negativen ausgegebenen Zahlen das korrekte "Vorzeichen" (-) nachgestellt ist.

Beispiel 6 (Operand überschreitet Vorgabe der Druckmaske):

```
10 LET A = 5678.23
20 PRINT USING "##.##"; A
RUN
Over flow error in 20
Ready
```

Erläuterungen zu Beispiel 6:

Enthält der Operand links vom Dezimalpunkt mehr Stellen als von der Druckmaske vorgesehen sind, dann wird die Verarbeitung mit der Fehlermeldung

```
Over flow error in ..
Ready
```

unterbrochen.

PE 26.2

Ergänzen Sie die fehlenden Ausgabedaten der folgenden Programme!

(1)
```
10 PRINT "          11111111112"
20 PRINT "01234567890123456789 0"
30 FOR X = 1 TO 5
40 READ N
50 PRINT USING "####.##"; N
60 NEXT X
70 DATA 1234, 1234.56, 12.346
80 DATA .5, 1.678
RUN
          11111111112
012345678901234567890

..............................

..............................

..............................

..............................

..............................

Ready
```

(2)

```
10 PRINT "          1111111112"
20 PRINT "01234567890123456789O"
30 LET A = 12.345
40 LET B = 24.681
50 PRINT USING "####.##"; A;
60 PRINT TAB(10) USING "####.##"; B
RUN
          1111111112
01234567890123456789O

..............................

Ready
```

(2)

```
10 LET A = 15
20 LET B = -25
30 PRINT USING "+##.##"; A
40 PRINT USING "+##.##"; B
50 PRINT USING "##.##+"; A
60 PRINT USING "##.##+"; B
70 PRINT USING "##.##-"; A
80 PRINT USING "##.##-"; B
RUN

..............................

..............................

..............................

..............................

..............................

..............................

Ready
```

LÖS 26.2

Ausgabedaten zu den Programmen der **PE 26.2:**

(1)
```
.
.
RUN
          11111111112
012345678901234567890
1234.00
1234.56
  12.35
   0.50
   1.68
Ready
```

(2)
```
.
.
RUN
          11111111112
012345678901234567890
  12.34      24.68
Ready
```

(3)
```
.
.
RUN
+15.00
-25.00
15.00+
25.00-
15.00
25.00-
Ready
```

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur

→ **LE 26.2** !

LE 26.3

Die folgenden Druckmaskenzeichen sind im Regelfall nicht so wichtig wie diejenigen, die wir in der vorangegangenen **LE 26.2** vorgestellt haben. Es ist daher nicht unbedingt erforderlich, daß Sie sich ihre Wirkungsweise einprägen. Darüber können Sie sich im Bedarfsfalle in dieser **LE 26.3** informieren!

Beispiel 1 (Druckmaskenzeichen **):

```
10 FOR X = 1 TO 4
20 READ N
30 PRINT USING "**##.##"; N
40 NEXT X
50 DATA 1.256, .2, 123.75, 1234.56
RUN
***1.26
***0.20
*123.75
1234.56
Ready
```

Erläuterungen zu Beispiel 1:

Die Zeichen ** (zwei Sterne) zu Beginn einer Druckmaske bewirken, daß die führenden Leerstellen eines numerischen Datenfeldes durch Sterne ersetzt werden (zum Zwecke der Schecksicherung etc.). Zugleich repräsentieren sie Druckstellen für zwei Ziffern.

B e i s p i e l 2 (Druckmaskenzeichen $$):

```
10 LET B = 17.56
20 PRINT USING "$$####.##"; B
RUN
   $17.56
Ready
```

Erläuterungen zu Beispiel 2:

Die Zeichen $$ zu Beginn einer Druckmaske haben zur Folge, daß unmittelbar links von den ausgegebenen Ziffern das $ -Zeichen erscheint. Zugleich repräsentierten $$ Druckstellen für zwei Ziffern. A c h t u n g : Im Zusammenhang mit $$ ist die Verwendung des Gleitkommaformats unzulässig, und + und - dürfen nur am Ende der Druckmaske erscheinen.

B e i s p i e l 3 (Druckmaskenzeichen ££):

```
10 LET B = 17.56
20 PRINT USING "££####.##"; B
RUN
   £17.56
Ready
```

Erläuterungen zu Beispiel 3:

Die "Erläuterungen zu Beispiel 2" gelten hier in analoger Weise.

B e i s p i e l 4 (Druckmaskenzeichen **$):

```
10 LET B = 17.56
20 PRINT USING "**$####.##"; B
RUN
****$17.56
Ready
```

Erläuterungen zu Beispiel 4:

Die Zeichen **$ zu Beginn einer Druckmaske verbinden die Wirkung von ** (vgl. Beispiel 1) und $$ (vgl. Beispiel 2). Zugleich repräsentieren **$ Druckstellen für drei Ziffern.

Beispiel 5 (Druckmaskenzeichen **£):

```
10 LET B = 17.56
20 PRINT USING "**£####.##"; B
RUN
****£17.56
Ready
```

Erläuterungen zu Beispiel 5:

Die "Erläuterungen zu Beispiel 4" gelten hier in analoger Weise.

Beispiel 6 (Druckmaskenzeichen ,):

```
10 LET C = 123456.78
20 LET D = 175.18
30 PRINT USING "###,###.##"; C
40 PRINT USING "###.##,"; D
RUN
123,456.78
175.18,
Ready
```

Erläuterungen zu Beispiel 6:

(1) Zum besseren Verständnis muß erwähnt werden, daß in den USA das Komma und der Punkt im Zusammenhang mit Dezimalzahlen genau umgekehrt verwendet werden wie in Deutschland.
(Beispiel für die Schreibweise eines Dollarbetrags
in Deutschland: $985.456,73 und
in den USA: $985,456.73)

Erscheint in einer Druckmaske links vom Dezimalpunkt an einer beliebigen Stelle ein Komma, so bewirkt dies (links vom Dezimalpunkt) die Ausgabe eines Kommas links von jeder dritten Ziffer, wenn links vom Komma eine weitere Ziffer folgt.

(2) Ein Komma am Ende der Druckmaske erzeugt hingegen nur ein Komma rechts von der ausgedruckten Zahl.

Beispiel 7 (Druckmaskenzeichen ↑↑↑↑):

```
10 LET A = 175.218
20 PRINT USING "##.##↑↑↑↑"; A
RUN
 1.75E02
Ready
```

Erläuterungen zu Beispiel 7:

Die vier Exponentialzeichen ↑↑↑↑ am Ende der Druckmaske bewirken, daß der Operand im Gleitkommaformat ausgegeben wird. Das erste Druckmaskenzeichen (#) ist in diesem Fall für das Vorzeichen der auszugebenden Zahl reserviert.

Beispiel 8 (Ausgabe von mehreren Operanden mit einer PRINT USING-Anweisung):

```
10 PRINT "          1111111111222222"
20 PRINT "0123456789012345678901234 5"
30 LET A = 16.7
40 LET B = 456.468
50 PRINT USING "####.##"; A; B
60 PRINT USING "####.##"; A, B
RUN
          1111111111222222
0123456789012345678901234 5
  16.70 456.47
  16.70 456.47
Ready
```

Erläuterungen zu Beispiel 8:

Nach einer PRINT USING-Anweisung darf man mehrere durch Semikola oder Kommata voneinander getrennte Operanden aufführen, die dann alle unmittelbar hintereinander entsprechend der Druckmaske ausgegeben werden. (Achtung: Das Komma bewirkt hier also keine Ausgabe zu Beginn der nächsten, vortabulierten Stelle.)

Beispiel 9 (Druckmaskenzeichen für Strings):

```
10 REM FORMATIERUNG VON STRINGS
20 REM ------------------------
30 LET A$ = "GUENTER O. HAMANN"
40 PRINT USING "!"; A$
50 PRINT USING "&     &; A$
60 PRINT USING "VORNAME: &     &"; A$
RUN
G
GUENTER
VORNAME: GUENTER
Ready
```

Erläuterungen zu Beispiel 9:

(1) Eine Druckmaske mit einem Ausrufezeichen (!) bewirkt, daß nur das erste Zeichen des Strings ausgegeben wird.

(2) Besteht die Druckmaske nur aus && , dann druckt das System die ersten zwei Zeichen des Strings. Jede Leerstelle zwischen && hat die Ausgabe eines zusätzlichen Zeichens zur Folge.

(3) Sind in der Druckmaske irgendwelche anderen als die erläuterten Zeichen enthalten, so werden diese zusammen mit dem auszugebenden Wert / String angezeigt.

Beispiel 10 (Druckmaske in einer String-Variablen / Kombination von PRINT und PRINT USING):

```
10 REM DRUCKMASKE IN EINER
20 REM STRING-VARIABLEN
30 REM ------------------
40 LET A$ = "DIE ZAHL LAUTET ####.##"
50 LET A = 1786
60 PRINT USING A$; A
70 REM KOMBINATION VON PRINT UND
80 REM PRINT USING
90 REM ------------------------
100 LET A = 17.35
110 LET B = 99.65
120 LET C = 44
```

```
130 PRINT "GELDBETRAEGE:"
140 PRINT A; B; USING "###.##"; C
RUN
DIE ZAHL LAUTET 1786.00
GELDBETRAEGE:
 17.35 99.65 44.00
Ready
```

Erläuterungen zu Beispiel 10:

(1) Es ist zulässig, eine Druckmaske (ggf. mit zusätzlichem Text) einer String-Variablen zuzuordnen und diese dann im Rahmen der PRINT USING-Anweisung zu verwenden.

(2) Normalerweise folgt bei PRINT USING das Wort USING unmittelbar nach PRINT. Man kann aber auch nach PRINT zunächst numerische Variablen aufführen, die unformatiert ausgegeben werden sollen, um dann nur jener (jenen) Variablen USING "..." voranzustellen, für die ein bestimmtes Ausgabeformat gewünscht wird.

PE 26.3

(Zur Lösung der folgenden Aufgaben dürfen Sie auf die Ausführungen in **LE 26.3** "zurückgreifen"!)

Ergänzen Sie die fehlenden Ausgabedaten der folgenden Programme!

(1)

```
10 FOR X = 1 TO 5
20 READ A
30 PRINT USING "**####.##"; A
40 NEXT X
50 DATA 1.2, 23.45, 567.899
60 DATA 1234, 123456.07
RUN

..............................

..............................

..............................

..............................

..............................

Ready
```

(2)
```
10 LET A = 123456.78
20 PRINT USING "###,###.##"; A
RUN

..............................

Ready
```

(3)
```
10 PRINT "          11111111112"
20 PRINT "012345678901234567890"
30 LET A$ = "SUMME: ###,###.##"
40 LET S = 1745.18
50 PRINT USING A$; S
RUN
          11111111112
012345678901234567890

..............................

Ready
```

LÖS 26.3

Ausgabedaten zu den Programmen der **PE 26.3**:

(1)

```
.
.
RUN
*****1.20
****23.45
***567.90
**1234.00
123456.07
Ready
```

(2)

```
.
.
RUN
123,456.78
Ready
```

(3)

```
.
.
RUN
          11111111112
012345678901234567890
SUMME:    1,745.18
Ready
```

Sollten Ihre Lösungen nicht richtig sein, ist es trotzdem nicht unbedingt erforderlich, die **LE 26.3** nochmals durchzuarbeiten. Informieren Sie sich besser im Bedarfsfall!

ÜBUNGsprogramm zu Lektion 26

Hinweise zum Programm (Schach):

(1) Die Realisierung eines Schachprogramms in einer Interpretersprache wie BASIC ist sehr problematisch, da die Ausführungszeiten unzumutbar lang werden.

(2) Das folgende Programm, das durch eine Veröffentlichung von A. H. Witt ("Leidenschaft am Krankenbett", in: microComputer-Welt, Nr. 3 / 15. Dezember 1981, Seite 24) angeregt wurde[1], erhebt weder Anspruch darauf, fehlerfrei zu sein, noch eine adäquate Lösung aufzuzeigen. Es will dem Leser lediglich eine kleine Anregung bieten, sich mit diesem interessanten Gebiet zu beschäftigen.[2]

(3) Zahlreiche Literaturhinweise zur Schachprogrammierung findet man in folgender Veröffentlichung:
Wacker, Detlef: "Tips für Schach-Programme", in: Computer Persönlich, Nr. 5 v. 23. 2. 1983, Seite 80 / 81
Dieser Autor empfiehlt zur Einarbeitung in die Thematik das folgende Werk, das inzwischen auch als preiswerte Taschenbuchausgabe vorliegt:
Hans P. Ketterling / Frieder Schwenkel / Ossi Weiner: "Schach dem Computer", Goldmann-Taschenbuch 10 861, München 1983, ISBN: 3-442-10 861-6

---

1) Unser Dank gebührt Herrn stud. rer. pol. Rolf Gross, der das aufgeführte Programm für den SHARP MZ-700 erstellte.

2) Wir bitten darum, daß jene Leser, die sich zu einer intensiveren Beschäftigung mit der Schachprogrammierung in BASIC entschlossen haben, uns ihre Anregungen und Verbesserungsvorschläge mitteilen.
Kontaktadresse: Prof. Dr. Günter O. Hamann
Fachhochschule Wilhelmshaven
Friedrich-Paffrath-Str. 101
2940 Wilhelmshaven

Programmliste:

```
10 REM *******************************
20 REM ******  SCHACH-BASIC  *********
30 REM *******************************
40 REM ******  TERMINIERUNG  *********
50 REM *******************************
60 DIM F$(8,8),M$(8,8),W1(8,8)
70 DIM D1$(10),D2$(10),D1(10),D2(10)
80 F1$=CHR$(200)+CHR$(200)+CHR$(200)
90 F2$="   "
100 CL$=CHR$(22)
110 FOR B=1 TO 8 : FOR Z=3 TO 6
120 F$(Z,B)=" "
130 NEXT Z : NEXT B
140 FOR B=1 TO 8
150 F$(2,B)="B" : F$(7,B)="b"
160 NEXT B
170 FOR B=1 TO 8
180 READ A$
190 F$(1,B)=A$
200 READ A$
210 F$(8,B)=A$
220 NEXT B
230 DATAT,t,S,s,L,l,D,d,K,k,L,l,S,s,T,t
240 F$(4,4)="B" : F$(2,4)=" "
250 REM *****************************
260 REM *****  HAUPTPROGRAMM *********
270 REM *****************************
280 GOSUB 2500
290 PRINT : PRINT
300 E$=" " : B1=0 : Z1=0 : B2=0 : Z2=0
310 BA$=" " : BN$=" "
320 PRINT" Wenn Sie Ihre Figur x";
330 PRINT" von C 5 nach D 6 "
340 PRINT" ziehen wollen, geben Sie";
350 PRINT" x,C,5,D,6 ein."
360 INPUT" Ihr Zug: ";E$,BA$,Z1,BN$,Z2
370 PRINT
380 B1=ASC(BA$)-64
390 B2=ASC(BN$)-64
400 F$(Z2,B2)=E$ : F$(Z1,B1)=" "
410 GOSUB 2500
420 PRINT
430 PRINT" Einen Moment bitte ! Stellung
swerte :"
```

```
440 PRINT
450 REM ******************************
460 REM ****  BERECHNUNG DES **********
470 REM ****  COMPUTERZUGES  **********
480 REM ******************************
490 FOR Z=1 TO 8 : FOR B=1 TO 8
500 M$(Z,B)=F$(Z,B) : W1(Z,B)=0
510 NEXT B : NEXT Z
520 SL=-5000
530 FOR Z=1 TO 8 : FOR B=1 TO 8
540 IF ASC(M$(Z,B))<100THEN W1(Z,B)=1
550 IF ASC(M$(Z,B))=32 THEN W1(Z,B)=0
560 IF ASC(M$(Z,B))>100THEN W1(Z,B)=2
570 NEXT B : NEXT Z
580 FOR Z=1 TO 8 : FOR B=1 TO 8
590 IF W1(Z,B)<>1 GOTO 2370
600 REM ******************************
610 REM ****  BAUERNZUEGE  ***********
620 REM ******************************
630 IF F$(Z,B)<>"B" GOTO 800
640 IF F$(Z+1,B)<>" " GOTO 720
650 Z1=Z+1 : B1=B
660 GOSUB 2740
670 IF Z<>2 GOTO 720
680 IF F$(Z+2,B)<>" " GOTO 720
690 Z1=Z+2 : B1=B
700 GOSUB 2740
710 GOTO 720
720 IF B=1 GOTO 760
730 IF W1(Z+1,B-1)<>2 GOTO 760
740 Z1=Z+1 : B1=B-1
750 GOSUB 2740
760 IF B=8 GOTO 800
770 IF W1(Z+1,B+1)<>2 GOTO 800
780 Z1=Z+1 : B1=B+1
790 GOSUB 2740
800 REM ******************************
810 REM ****  SPRINGERZUEGE  **********
820 REM ******************************
830 IF F$(Z,B)<>"S" GOTO 1240
840 IF (B>1)*(Z>2) GOTO 860
850 GOTO 890
```

```
860 IF W1(Z-2,B-1)=1 GOTO 890
870 Z1=Z-2 : B1=B-1
880 GOSUB 2740
890 IF (B>2)*(Z>1) GOTO 910
900 GOTO 940
910 IF W1(Z-1,B-2)=1 GOTO 940
920 Z1=Z-1 : B1=B-2
930 GOSUB 2740
940 IF (B>2)*(Z<8) GOTO 960
950 GOTO 990
960 IF W1(Z+1,B-2)=1 GOTO 990
970 Z1=Z+1 : B1=B-2
980 GOSUB 2740
990 IF (B>1)*(Z<7) GOTO 1010
1000 GOTO 1040
1010 IF W1(Z+2,B-1)=1 GOTO 1040
1020 Z1=Z+2 : B1=B-1
1030 GOSUB 2740
1040 IF (B<8)*(Z<7) GOTO 1060
1050 GOTO 1090
1060 IF W1(Z+2,B+1)=1 GOTO 1090
1070 Z1=Z+2 : B1=B+1
1080 GOSUB 2740
1090 IF (B<7)*(Z<8) GOTO 1110
1100 GOTO 1140
1110 IF W1(Z+1,B+2)=1 GOTO 1140
1120 Z1=Z+1 : B1=B+2
1130 GOSUB 2740
1140 IF (B<7)*(Z>1) GOTO 1160
1150 GOTO 1190
1160 IF W1(Z-1,B+2)=1 GOTO 1190
1170 Z1=Z-1 : B1=B+2
1180 GOSUB 2740
1190 IF (B<8)*(Z>2) GOTO 1210
1200 GOTO 1240
1210 IF W1(Z-2,B+1)=1 GOTO 1240
1220 Z1=Z-2 : B1=B+1
1230 GOSUB 2740
1240 REM *****************************
1250 REM ***  DIAGONALZUEGE  **********
1260 REM *****************************
1270 IF F$(Z,B)="L" GOTO 1300
```

```
1280 IF F$(Z,B)="D" GOTO 1300
1290 GOTO 1680
1300 IF Z>B THEN C=Z
1310 IF Z<=B THEN C=B
1320 IF C=8 GOTO 1390
1330 FOR H=1 TO (8-C)
1340 IF W1(Z+H,B+H)=1 GOTO 1390
1350 Z1=Z+H : B1=B+H
1360 GOSUB 2740
1370 IF W1(Z+H,B+H)=2 GOTO 1390
1380 NEXT H
1390 IF(B=8)+(Z=1) GOTO 1490
1400 C=9-Z
1410 IF B>C THEN D=B
1420 IF B<=C THEN D=C
1430 FOR H=1 TO (8-D)
1440 IF W1(Z-H,B+H)=1 GOTO 1490
1450 Z1=Z-H : B1=B+H
1460 GOSUB 2740
1470 IF W1(Z-H,B+H)=2 THEN 1490
1480 NEXT H
1490 IF(B=1)+(Z=1) GOTO 1580
1500 IF B>Z THEN C=Z
1510 IF B<=Z THEN C=B
1520 FOR H=1 TO C
1530 IF W1(Z-H,B-H)=1 GOTO 1580
1540 Z1=Z-H : B1=B-H
1550 GOSUB 2740
1560 IF W1(Z-H,B-H)=2 GOTO 1580
1570 NEXT H
1580 IF (Z=8)+(B=1) GOTO 1680
1590 C=9-Z
1600 IF Z>C THEN D=Z
1610 IF Z<=C THEN D=C
1620 FOR H=1 TO (8-D)
1630 IF W1(Z+H,B-H)=1 GOTO 1680
1640 Z1=Z+H : B1=B-H
1650 GOSUB 2740
1660 IF W1(Z+H,B-H)=2 GOTO 1680
1670 NEXT H
```

```
1680 REM ****************************
1690 REM *** HORIZONTAL/VERTIKALZUEGE *
1700 REM ****************************
1710 IF F$(Z,B)="T" GOTO 1740
1720 IF F$(Z,B)="D" GOTO 1740
1730 GOTO 2020
1740 IF B=8 GOTO 1810
1750 FOR H=B+1 TO 8
1760 IF W1(Z,H)=1 GOTO 1810
1770 Z1=Z : B1=H
1780 GOSUB 2740
1790 IF W1(Z,H)=2 GOTO 1810
1800 NEXT H
1810 IF B=1 GOTO 1880
1820 FOR H=B-1 TO 1 STEP-1
1830 IF W1(Z,H)=1 GOTO 1880
1840 Z1=Z : B1=H
1850 GOSUB 2740
1860 IF W1(Z,H)=2 GOTO 1880
1870 NEXT H
1880 IF Z=8 GOTO 1950
1890 FOR H=Z+1 TO 8
1900 IF W1(H,B)=1 GOTO 1950
1910 Z1=H : B1=B
1920 GOSUB 2740
1930 IF W1(H,B)=2 GOTO 1950
1940 NEXT H
1950 IF Z=1 GOTO 2020
1960 FOR H=Z-1 TO 1 STEP-1
1970 IF W1(H,B)=1 GOTO 2020
1980 Z1=H : B1=B
1990 GOSUB 2740
2000 IF W1(H,B)=2 GOTO 2020
2010 NEXT H
2020 REM ****************************
2030 REM *** KOENIGSZUEGE ***********
2040 REM ****************************
2050 IF F$(Z,B)<>"K" GOTO 2370
2060 IF B=8 GOTO 2180
2070 IF W1(Z,B+1)=1 GOTO 2100
2080 Z1=Z : B1=B+1
2090 GOSUB 2740
```

```
2100 IF Z=8 GOTO 2140
2110 IF W1(Z+1,B+1)=1 GOTO 2140
2120 Z1=Z+1 : B1=B+1
2130 GOSUB 2740
2140 IF Z=1 GOTO 2180
2150 IF W1(Z-1,B+1)=1 GOTO 2180
2160 Z1=Z-1 : B1=B+1
2170 GOSUB 2740
2180 IF Z=8 GOTO 2220
2190 IF W1(Z+1,B)=1 GOTO 2220
2200 Z1=Z+1 : B1=B
2210 GOSUB 2740
2220 IF Z=1 GOTO 2250
2230 IF W1(Z-1,B)=1 GOTO 2250
2240 Z1=Z-1 : B1=B
2250 IF B=1 GOTO 2370
2260 IF Z=8 GOTO 2300
2270 IF W1(Z+1,B-1)=1 GOTO 2300
2280 Z1=Z+1 : B1=B-1
2290 GOSUB 2740
2300 IF W1(Z,B-1)=1 GOTO 2330
2310 Z1=Z : B1=B-1
2320 GOSUB 2740
2330 IF Z=1 GOTO 2370
2340 IF W1(Z-1,B-1)=1 GOTO 2370
2350 Z1=Z-1 : B1=B-1
2360 GOSUB 2740
2370 NEXT B
2380 NEXT Z
2390 F$(Y1,X1)=" " : F$(Y2,X2)=X$
2400 M$(Y1,X1)=" " : M$(Y2,X2)=X$
2410 L$=""
2420 X1$=CHR$(X1+64) : X2$=CHR$(X2+64)
2430 PRINT
2440 PRINT"Mein Zug: ";X$;",";X1$;Y1;","
X2$;Y2;"; CR-Taste druecken!"
2450 GET L$ : IF L$="" GOTO 2450
2460 GOTO 280
2470 REM ****************************
2480 REM ****  HAUPTPROGRAMM ENDE  ****
2490 REM ****************************
```

```
2500 REM ******************************
2510 REM ****  OUTPUT UNTERPROGRAMM  **
2520 REM ******************************
2530 PRINTCL$
2540 CURSOR 0,0
2550 PRINT"   A  B  C  D  E  F  G  H "
2560 FOR Z=4 TO 1 STEP-1
2570 FOR A=1 TO 2
2580 PRINT2*Z;F1$;F2$;F1$;F2$;F1$;F2$;F1
$;F2$
2590 NEXT A
2600 FOR B=1 TO 2
2610 PRINT2*Z-1;F2$;F1$;F2$;F1$;F2$;F1$;
F2$;F1$
2620 NEXT B
2630 NEXT Z
2640 FOR Z=8 TO 1 STEP-1
2650 FOR B=1 TO 8
2660 Y=(8-Z)*2+2
2670 X=(B-1)*3+3
2680 CURSOR X,Y
2690 IF F$(Z,B)="  " GOTO 2710
2700 PRINTF$(Z,B)
2710 NEXT B
2720 NEXT Z
2730 RETURN
2740 REM ******************************
2750 REM ***  UP-STELLUNGSBEWERTUNG  **
2760 REM ******************************
2770 IF F$(Z1,B1)="b" THEN LET SP=5
2780 IF F$(Z1,B1)="s" THEN LET SP=30
2790 IF F$(Z1,B1)="!" THEN LET SP=50
2800 IF F$(Z1,B1)="t" THEN LET SP=60
2810 IF F$(Z1,B1)="d" THEN LET SP=100
2820 IF F$(Z1,B1)="k" THEN LET SP=101
2830 M$(Z1,B1)=M$(Z,B) : M$(Z,B)="  "
2840 V=W1(Z1,B1):W1(Z1,B1)=1:W1(Z,B)=0
2850 QS=0 : SW=0
2860 FOR Z2=1 TO 8 : FOR B2=1 TO 8
2870 IF W1(Z2,B2)=1 GOTO 2890
2880 GOTO 5960
2890 Y=Z2 : X=B2
```

```
2900 REM ******************************
2910 REM ** DECKENDE/DROHENDE FIGUREN *
2920 REM ******************************
2930 FOR I=1 TO 10 : D1$(I)=" "
2940 D2$(I)=" " : NEXT I
2950 R=1 : P=1
2960 REM ******************************
2970 REM ** HORIZONTAL/VERTIKAL *******
2980 REM ******************************
2990 IF X=1 GOTO 3090
3000 FOR G=(X-1) TO 1 STEP-1
3010 IF W1(Y,G)<>0 GOTO 3030
3020 NEXT G
3030 IF M$(Y,G)="T" THEN D1$(R)="T"
3040 IF M$(Y,G)="D" THEN D1$(R)="D"
3050 IF D1$(R)<>" " THEN R=R+1
3060 IF M$(Y,G)="t" THEN D2$(P)="t"
3070 IF M$(Y,G)="d" THEN D2$(P)="d"
3080 IF D2$(P)<>" " THEN P=P+1
3090 IF X=8 GOTO 3200
3100 G=X
3110 G=G+1
3120 IF W1(Y,G)<>0 GOTO 3140
3130 IF G<>8 GOTO 3110
3140 IF M$(Y,G)="T" THEN D1$(R)="T"
3150 IF M$(Y,G)="D" THEN D1$(R)="D"
3160 IF D1$(R)<>" " THEN R=R+1
3170 IF M$(Y,G)="t" THEN D2$(P)="t"
3180 IF M$(Y,G)="d" THEN D2$(P)="d"
3190 IF D2$(P)<>" " THEN P=P+1
3200 IF Y=1 GOTO 3300
3210 FOR G=(Y-1) TO 1 STEP-1
3220 IF W1(G,X)<>0 GOTO 3240
3230 NEXT G
3240 IF M$(G,X)="T" THEN D1$(R)="T"
3250 IF M$(G,X)="D" THEN D1$(R)="D"
3260 IF D1$(R)<>" " THEN R=R+1
3270 IF M$(G,X)="t" THEN D2$(P)="t"
3280 IF M$(G,X)="d" THEN D2$(P)="d"
3290 IF D2$(P)<>" " THEN P=P+1
3300 IF Y=8 GOTO 3410
3310 G=Y
```

```
3320 G=G+1
3330 IF W1(G,X)<>0 GOTO 3350
3340 IF G<>8 GOTO 3320
3350 IF M$(G,X)="T" THEN D1$(R)="T"
3360 IF M$(G,X)="D" THEN D1$(R)="D"
3370 IF D1$(R)<>" " THEN R=R+1
3380 IF M$(G,X)="t" THEN D2$(P)="t"
3390 IF M$(G,X)="d" THEN D2$(P)="d"
3400 IF D2$(P)<>" " THEN P=P+1
3410 REM ****************************
3420 REM ** DIAGONAL ****************
3430 REM ****************************
3440 IF Y>X THEN C=Y
3450 IF Y<=X THEN C=X
3460 IF C=8 GOTO 3570
3470 G=0
3480 G=G+1
3490 IF W1(Y+G,X+G)<>0 GOTO 3510
3500 IF G<>8-C GOTO 3480
3510 IF M$(Y+G,X+G)="L" THEN D1$(R)="L"
3520 IF M$(Y+G,X+G)="D" THEN D1$(R)="D"
3530 IF D1$(R)<>" " THEN R=R+1
3540 IF M$(Y+G,X+G)="!" THEN D2$(P)="!"
3550 IF M$(Y+G,X+G)="d" THEN D2$(P)="d"
3560 IF D2$(P)<>" " THEN P=P+1
3570 C=9-Y
3580 IF (X=8)+(Y=1) GOTO 3710
3590 IF X>C THEN D=X
3600 IF X<=C THEN D=C
3610 G=0
3620 FOR G=0 TO (8-D)
3630 IF W1(Y-G,X+G)<>0 GOTO 3650
3640 NEXT G
3650 IF M$(Y-G,B+G)="L" THEN D1$(R)="L"
3660 IF M$(Y-G,X+G)="D" THEN D1$(R)="D"
3670 IF D1$(R)<>" " THEN R=R+1
3680 IF M$(Y-G,X+G)="!" THEN D2$(P)="|"
3690 IF M$(Y-G,X+G)="d" THEN D2$(P)="d"
3700 IF D2$(P)<>" " THEN P=P+1
3710 IF X>Y THEN C=Y
3720 IF X<=Y THEN C=X
3730 IF (X=1)+(Y=1) GOTO 3830
```

```
3740 FOR G=1 TO (C-1)
3750 IF W1(Y-G,X-G)<>0 GOTO 3770
3760 NEXT G
3770 IF M$(Y-G,X-G)="L" THEN D1$(R)="L"
3780 IF M$(Y-G,X-G)="D" THEN D1$(R)="D"
3790 IF D1$(R)<>" " THEN R=R+1
3800 IF M$(Y-G,X-G)="!" THEN D2$(P)="!"
3810 IF M$(Y-G,X-G)="d" THEN D2$(P)="d"
3820 IF D2$(P)<>" " THEN P=P+1
3830 IF (Y=8)+(X=1) GOTO 3970
3840 C=9-X
3850 IF Y>C THEN D=Y
3860 IF Y<=C THEN D=C
3870 G=0
3880 G=G+1
3890 IF W1(Y+G,X-G)<>0 GOTO 3910
3900 IF G<>(8-D) GOTO 3880
3910 IF M$(Y+G,X-G)="L" THEN D1$(R)="L"
3920 IF M$(Y+G,X-G)="D" THEN D1$(R)="D"
3930 IF D1$(R)<>" " THEN R=R+1
3940 IF M$(Y+G,X-G)="!" THEN D2$(P)="!"
3950 IF M$(Y+G,X-G)="d" THEN D2$(P)="d"
3960 IF D2$(P)<>" " THEN P=P+1
3970 REM *****************************
3980 REM ** SPRINGER *****************
3990 REM *****************************
4000 IF (X>1)*(Y>2) GOTO 4020
4010 GOTO 4060
4020 IF M$(Y-2,X-1)="S" THEN D1$(R)="S"
4030 IF D1$(R)<>" " THEN R=R+1
4040 IF M$(Y-2,X-1)="s" THEN D2$(P)="s"
4050 IF D2$(P)<>" " THEN P=P+1
4060 IF (X>2)*(Y>1) GOTO 4080
4070 GOTO 4120
4080 IF M$(Y-1,X-2)="S" THEN D1$(R)="S"
4090 IF D1$(R)<>" " THEN R=R+1
4100 IF M$(Y-1,X-2)="s" THEN D2$(P)="s"
4110 IF D2$(P)<>" " THEN P=P+1
4120 IF (X>2)*(Y<8) GOTO 4140
4130 GOTO 4180
4140 IF M$(Y+1,X-2)="S" THEN D1$(R)="S"
4150 IF D1$(R)<>" " THEN R=R+1
```

```
4160 IF M$(Y+1,X-2)="s" THEN D2$(P)="s"
4170 IF D2$(P)<>" " THEN P=P+1
4180 IF (X>1)*(Y<7) GOTO 4200
4190 GOTO 4240
4200 IF M$(Y+2,X-1)="S" THEN D1$(R)="S"
4210 IF D1$(R)<>" " THEN R=R+1
4220 IF M$(Y+2,X-1)="s" THEN D2$(P)="s"
4230 IF D2$(P)<>" " THEN P=P+1
4240 IF (X<8)*(Y<7) GOTO 4260
4250 GOTO 4300
4260 IF M$(Y+2,X+1)="S" THEN D1$(R)="S"
4270 IF D1$(R)<>" " THEN R=R+1
4280 IF M$(Y+2,X+1)="s" THEN D2$(P)="s"
4290 IF D2$(P)<>" " THEN P=P+1
4300 IF (X<7)*(Y<8) GOTO 4320
4310 GOTO 4360
4320 IF M$(Y+1,X+2)="S" THEN D1$(R)="S"
4330 IF D1$(R)<>" " THEN R=R+1
4340 IF M$(Y+1,X+2)="s" THEN D2$(P)="s"
4350 IF D2$(P)<>" " THEN P=P+1
4360 IF (X<7)*(Y>1) GOTO 4380
4370 GOTO 4480
4380 IF M$(Y-1,X+2)="S" THEN D1$(R)="S"
4390 IF D1$(R)<>" " THEN R=R+1
4400 IF M$(Y-1,X+2)="s" THEN D2$(P)="s"
4410 IF D2$(P)<>" " THEN P=P+1
4420 IF (X<8)*(Y>2) GOTO 4440
4430 GOTO 4480
4440 IF M$(Y-2,X+1)="S" THEN D1$(R)="S"
4450 IF D1$(R)<>" " THEN R=R+1
4460 IF M$(Y-2,X+1)="s" THEN D2$(P)="s"
4470 IF D2$(P)<>" " THEN P=P+1
4480 REM ****************************
4490 REM ** KOENIG/BAUERN ************
4500 REM ****************************
4510 IF X=8 GOTO 4680
4520 IF M$(Y,X+1)="K" THEN D1$(R)="K"
4530 IF D1$(R)<>" " THEN R=R+1
4540 IF M$(Y,X+1)="k" THEN D2$(P)="k"
4550 IF D2$(P)<>" " THEN P=P+1
4560 IF Y=8 GOTO 4620
4570 IF M$(Y+1,X+1)="K" THEN D1$(R)="K"
```

```
4580 IF D1$(R)<>" " THEN R=R+1
4590 IF M$(Y+1,X+1)="k" THEN D2$(P)="k"
4600 IF M$(Y+1,X+1)="b" THEN D2$(P)="b"
4610 IF D2$(P)<>" " THEN P=P+1
4620 IF Y=1 GOTO 4680
4630 IF M$(Y-1,X+1)="K" THEN D1$(R)="K"
4640 IF M$(Y-1,X+1)="B" THEN D1$(R)="B"
4650 IF D1$(R)<>" " THEN R=R+1
4660 IF M$(Y-1,X+1)="k" THEN D2$(P)="k"
4670 IF D2$(P)<>" " THEN P=P+1
4680 IF Y=8 GOTO 4730
4690 IF M$(Y+1,X)="K" THEN D1$(R)="K"
4700 IF D1$(R)<>" " THEN R=R+1
4710 IF M$(Y+1,X)="k" THEN D2$(P)="k"
4720 IF D2$(P)<>" " THEN P=P+1
4730 IF Y=1 GOTO 4780
4740 IF M$(Y-1,X)="K" THEN D1$(R)="K"
4750 IF D1$(R)<>" " THEN R=R+1
4760 IF M$(Y-1,X)="k" THEN D2$(P)="k"
4770 IF D2$(P)<>" " THEN P=P+1
4780 IF X=1 GOTO 4950
4790 IF Y=8 GOTO 4850
4800 IF M$(Y+1,X-1)="K" THEN D1$(R)="K"
4810 IF D1$(R)<>" " THEN R=R+1
4820 IF M$(Y+1,X-1)="k" THEN D2$(P)="k"
4830 IF M$(Y+1,X-1)="b" THEN D2$(P)="b"
4840 IF D2$(P)<>" " THEN P=P+1
4850 IF M$(Y,X-1)="K" THEN D1$(R)="K"
4860 IF D1$(R)<>" " THEN R=R+1
4870 IF M$(Y,X-1)="k" THEN D2$(P)="k"
4880 IF D2$(P)<>" " THEN P=P+1
4890 IF Y=1 GOTO 4950
4900 IF M$(Y-1,X-1)="K" THEN D1$(R)="K"
4910 IF M$(Y-1,X-1)="B" THEN D1$(R)="B"
4920 IF D1$(R)<>" " THEN R=R+1
4930 IF M$(Y-1,X-1)="k" THEN D2$(P)="k"
4940 IF D2$(P)<>" " THEN P=P+1
4950 REM ****************************
4960 REM ** DECKENDE/DROHENDE FIG/ENDE*
4970 REM ****************************
4980 REM ** D1$(R)=DECKENDE FIGUREN+1
4990 REM ** D2$(P)=DROHENDE FIGUREN+1
```

```
5000 REM ******************************
5010 REM *** SORTIEREN DER FIGUREN ****
5020 REM ******************************
5030 REM ******************************
5040 REM *** DECKENDE FIGUREN *********
5050 REM ******************************
5060 FOR A=1 TO 10 : D1(A)=0 : D2(A)=0
5070 NEXT A : S=2
5080 IF F$(Y,X)="B" THEN LET D1(1)=5
5090 IF F$(Y,X)="S" THEN LET D1(1)=30
5100 IF F$(Y,X)="L" THEN LET D1(1)=50
5110 IF F$(Y,X)="T" THEN LET D1(1)=60
5120 IF F$(Y,X)="D" THEN LET D1(1)=100
5130 IF F$(Y,X)="K" THEN LET D1(1)=101
5140 IF R=1 GOTO 5330
5150 FOR G=1 TO R
5160 IF D1$(G)="B" THEN D1(S)=5 : S=S+1
5170 NEXT G
5180 FOR G=1 TO R
5190 IF D1$(G)="S" THEN D1(S)=30: S=S+1
5200 NEXT G
5210 FOR G=1 TO R
5220 IF D1$(G)="L" THEN D1(S)=50: S=S+1
5230 NEXT G
5240 FOR G=1 TO R
5250 IF D1$(G)="T" THEN D1(S)=60: S=S+1
5260 NEXT G
5270 FOR G=1 TO R
5280 IF D1$(G)="D" THEN D1(S)=100:S=S+1
5290 NEXT G
5300 FOR G=1 TO R
5310 IF D1$(G)="K" THEN D1(S)=101:S=S+1
5320 NEXT G
5330 REM ******************************
5340 REM *** DROHENDE FIGUREN *********
5350 REM ******************************
5360 S=1 : P=P-1 : R=R-1
5370 IF P=0 GOTO 5580
5380 FOR G=1 TO P
5390 IF D2$(G)="b" THEN D2(S)=5 : S=S+1
5400 NEXT G
5410 FOR G=1 TO P
```

```
5420 IF D2$(G)="s" THEN D2(S)=30: S=S+1
5430 NEXT G
5440 FOR G=1 TO P
5450 IF D2$(G)="l" THEN D2(S)=50: S=S+1
5460 NEXT G
5470 FOR G=1 TO P
5480 IF D2$(G)="t" THEN D2(S)=60: S=S+1
5490 NEXT G
5500 FOR G=1 TO P
5510 IF D2$(G)="d" THEN D2(S)=100:S=S+1
5520 NEXT G
5530 FOR G=1 TO P
5540 IF D2$(G)="k" THEN D2(S)=101:S=S+1
5550 NEXT G
5560 REM ****************************
5570 REM *** BEDROHUNGSPUNKTE *********
5580 REM ****************************
5590 Q=0
5600 IF P=0 GOTO 5960
5610 IF(R=P)*(D2(P)<>101)*(D1(1)<>101)GO
TO 5630
5620 GOTO 5660
5630 FOR G=1 TO R
5640 Q=Q+D2(G)-D1(G)
5650 NEXT G
5660 IF(R>P)*(D2(1)<>101)*(D2(P)<>101)GO
TO 5680
5670 GOTO 5710
5680 FOR G=1 TO P
5690 Q=Q+D2(G)-D1(G)
5700 NEXT G
5710 IF(P>R)*(D1(1)<>101)*(D1(R+1)<>101)
GOTO 5730
5720 GOTO 5780
5730 IF R=0 GOTO 5770
5740 FOR G=1 TO R
5750 Q=Q+D2(G)-D1(G)
5760 NEXT G
5770 Q=Q-D1(R+1)
5780 IF(P>1)*(D2(P)=101)*(R>=P)GOTO5800
5790 GOTO 5830
5800 FOR G=1 TO (P-1)
```

```
5810 Q=Q+D2(G)-D1(G)
5820 NEXT G
5830 IF(P>R)*(D1(R+1)=101)*(R>1)GOTO 585
0
5840 GOTO 5900
5850 IF P=1 GOTO 5890
5860 FOR G=1 TO (R-1)
5870 Q=Q+D2(G)-D1(G)
5880 NEXT G
5890 Q=Q-D1(R)
5900 IF(D1(G)=101)*(P>=1)THEN Q=-1000
5910 REM *****************************
5920 REM *** BEDROHUNGSPUNKTE/ENDE ****
5930 REM *****************************
5940 IF Q>0 GOTO 5960
5950 QS=QS+Q
5960 NEXT B2
5970 NEXT Z2
5980 REM *****************************
5990 REM *** WIRKUNGSPUNKTE ***********
6000 REM *****************************
6010 W=0
6020 FOR Z2=1 TO 8 : FOR B2=1 TO 8
6030 IF W1(Z2,B2)<>1 GOTO 7280
6040 REM *****************************
6050 REM *** BAUERN  *****************
6060 REM *****************************
6070 IF M$(Z2,B2)<>"B" GOTO 6120
6080 IF B2=1 GOTO 6100
6090 IF W1(Z2+1,B2-1)<>0 THEN LET W=W+1
6100 IF B2=8 GOTO 6120
6110 IF W1(Z2+1,B2+1)<>0 THEN LET W=W+1
6120 REM *****************************
6130 REM *** SPRINGER  ***************
6140 REM *****************************
6150 IF M$(Z2,B2)<>"S" GOTO 6410
6160 IF (B2>1)*(Z2>2) GOTO 6180
6170 GOTO 6190
6180 IF W1(Z2-2,B2-1)<>0 THEN LET W=W+1
6190 IF (B2>2)*(Z2>1) GOTO 6210
6200 GOTO 6220
6210 IF W1(Z2-1,B2-2)<>0 THEN LET W=W+1
```

```
6220 IF (B2>2)*(Z2<8) GOTO 6240
6230 GOTO 6250
6240 IF W1(Z2+1,B2-2)<>0 THEN LET W=W+1
6250 IF (B2>1)*(Z2<7) GOTO 6270
6260 GOTO 6280
6270 IF W1(Z2+2,B2-1)<>0 THEN LET W=W+1
6280 IF (B2<8)*(Z2<7) GOTO 6300
6290 GOTO 6310
6300 IF W1(Z2+2,B2+1)<>0 THEN LET W=W+1
6310 IF (B2<7)*(Z2<8) GOTO 6330
6320 GOTO 6340
6330 IF W1(Z2+1,B2+2)<>0 THEN LET W=W+1
6340 IF (B2<7)*(Z2>1) GOTO 6360
6350 GOTO 6370
6360 IF W1(Z2-1,B2+2)<>0 THEN LET W=W+1
6370 IF (B2<8)*(Z2>2) GOTO 6390
6380 GOTO 6400
6390 IF W1(Z2-2,B2+1)<>0 THEN LET W=W+1
6400 REM ****************************
6410 REM *** HORIZONTALE/VERTIKALE  ***
6420 REM ****************************
6430 IF M$(Z2,B2)="T" GOTO 6460
6440 IF M$(Z2,B2)="D" GOTO 6460
6450 GOTO 6700
6460 IF B2=8 GOTO 6520
6470 FOR G=(B2+1) TO 8
6480 IF W1(Z2,G)=0 GOTO 6510
6490 W=W+1
6500 IF W1(Z2,G)<>0 GOTO 6520
6510 NEXT G
6520 IF B2=1 GOTO 6580
6530 FOR G=(B2-1) TO 1 STEP-1
6540 IF W1(Z2,G)=0 GOTO 6570
6550 W=W+1
6560 IF W1(Z2,G)<>0 GOTO 6580
6570 NEXT G
6580 IF Z2=8 GOTO 6640
6590 FOR G=(Z2+1) TO 8
6600 IF W1(G,B2)=0 GOTO 6630
6610 W=W+1
6620 IF W1(G,B2)<>0 GOTO 6640
6630 NEXT G
```

```
6640 IF Z2=1 GOTO 6700
6650 FOR G=(Z2-1) TO 1 STEP-1
6660 IF W1(G,B2)=0 GOTO 6690
6670 W=W+1
6680 IF W1(G,B2)<>0 GOTO 6700
6690 NEXT G
6700 REM ******************************
6710 REM *** DIAGONALE  ***************
6720 REM ******************************
6730 IF M$(Z2,B2)="L" GOTO 6760
6740 IF M$(Z2,B2)="D" GOTO 6760
6750 GOTO 7100
6760 IF Z2>B2 THEN C=Z2
6770 IF Z2<=B2 THEN C=B2
6780 IF C=8 GOTO 6840
6790 FOR G=1 TO (8-C)
6800 IF W1(Z2+G,B2+G)=0 GOTO 6830
6810 W=W+1
6820 IF W1(Z2+G,B2+G)<>0 GOTO 6840
6830 NEXT G
6840 C=9-Z2
6850 IF (B2=8)+(Z2=1) GOTO 6930
6860 IF B2>C THEN D=B2
6870 IF B2<=C THEN D=C
6880 FOR G=1 TO (8-D)
6890 IF W1(Z2-G,B2+G)=0 GOTO 6920
6900 W=W+1
6910 IF W1(Z2-G,B2+G)<>0 GOTO 6930
6920 NEXT G
6930 IF (B2=1)+(Z2=1) GOTO 7010
6940 IF B2>Z2 THEN C=Z2
6950 IF B2<=Z2 THEN C=B2
6960 FOR G=1 TO C
6970 IF W1(Z2-G,B2-G)=0 GOTO 7000
6980 W=W+1
6990 IF W1(Z2-G,B2-G)<>0 GOTO 7010
7000 NEXT G
7010 C=9-B2
7020 IF (Z2=8)+(B2=1) GOTO 7110
7030 IF Z2>C THEN D=Z2
7040 IF Z2<=C THEN D=C
7050 FOR G=1 TO (8-D)
```

```
7060 IF W1(Z2+G,B2-G)=0 GOTO 7090
7070 W=W+1
7080 IF W1(Z2+G,B2-G)<>0 GOTO 7100
7090 NEXT G
7100 REM *****************************
7110 REM *** KOENIG  *****************
7120 REM *****************************
7130 IF M$(Z2,B2)<>"K" GOTO 7280
7140 IF B2=8 GOTO 7200
7150 IF W1(Z2,B2+1)=1 THEN W=W+1
7160 IF Z2=8 GOTO 7180
7170 IF W1(Z2+1,B2+1)=1 THEN W=W+1
7180 IF Z2=1 GOTO 7200
7190 IF W1(Z2-1,B2+1)=1 THEN W=W+1
7200 IF W1(Z2-1,B2)=1 THEN W=W+1
7210 IF Z2=8 GOTO 7230
7220 IF W1(Z2+1,B2)=1 THEN W=W+1
7230 IF B2=1 GOTO 7280
7240 IF W1(Z2+1,B2-1)=1 THEN W=W+1
7250 IF W1(Z2,B2-1)=1 THEN W=W+1
7260 IF Z2=1 GOTO 7280
7270 IF W1(Z2-1,B2-1)=1 THEN W=W+1
7280 NEXT B2
7290 NEXT Z2
7300 REM *****************************
7310 REM *** WIRKUNGSPUNKTE/ENDE ******
7320 REM *****************************
7330 SW=QS+W
7340 PRINTSW;
7350 W1(Z1,B1)=V : W1(Z,B)=1
7360 SW=SW+SP : SP=0
7370 IF SW<=SL GOTO 7410
7380 X1=B : Y1=Z : X2=B1 : Y2=Z1
7390 X$=F$(Z,B)
7400 SL=SW
7410 FOR Z1=1 TO 8 : FOR B1=1 TO 8
7420 M$(Z1,B1)=F$(Z1,B1)
7430 NEXT B1
7440 NEXT Z1
7450 RETURN
7460 REM *****************************
7470 REM **** STELLUNGSBEWERTUNG/ENDE *
7480 REM *****************************
```

27. Datei-Befehle

LE 27.1

Da der Hauptspeicher eines Rechners kaum jemals über genügend Platz verfügt, um große Datenmengen und alle erstellten Programme zu speichern, benötigt man periphere Geräte (vgl. Lektion 2, insbesondere **LE 2.3** !), die an die Zentraleinheit angeschlossen werden. Im Zusammenhang mit ihnen ist die Behandlung von Dateien und das Wissen darüber unentbehrlich.

Was versteht man unter einer "D a t e i" ?

Für unsere Zwecke definieren wir den Begriff "D a t e i" (engl. "file") als einen B e s t a n d von s i n n v o l l z u s a m m e n - g e h ö r i g e n D a t e n, d e r a u f e i n e m P e r i p h e r i e - g e r ä t g e s p e i c h e r t i s t. Die Daten können einerseits Zahlen und/oder Strings sein und andererseits auch Programmbefehle.

PE 27.1

Welche der folgenden Behauptungen ist/sind richtig?

A Bei modernen Rechnern ist der Hauptspeicher groß genug, um alle denkbaren Datenmengen und alle erstellten Programme gleichzeitig aufnehmen zu können.

B Der Hauptspeicher eines Rechners verfügt kaum jemals über genügend Platz, um große Datenmengen und alle erstellten Programme aufzunehmen.

C Unter "D a t e i" (engl. "file") versteht man einen Datenbestand von sinnvoll zusammengehörigen Daten, der auf einem Peripheriegerät gespeichert ist.

D Die Daten einer Datei können einerseits Zahlen und/oder Strings sein und andererseits auch Programmbefehle.

E Eine Datei ist die kleinste Informationseinheit, die entweder den Wert "0" oder "1" annehmen kann.

Der/die Lösungsbuchstabe/n lautet/lauten

. . . .

LÖS 27.1

Die Lösungsbuchstaben lauten

B, C, D .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur
→ **LE 27.1** !

LE 27.2

Als Peripherie ist für den SHARP MZ-700 (u. a.) ein (Band-) Kassettengerät vorgesehen, das

- zum Laden des BASIC-Interpreters,
- zum Speichern und Laden von Anwenderprogrammen (= Programm-Dateien) und
- zum Beschreiben und Lesen von sequentiellen Daten-Dateien

benötigt wird.

Wie man den BASIC-Interpreter lädt, haben wir auf Seite 6-12 ff. ausführlich erläutert.

Auch das Speichern und Laden von Anwenderprogrammen haben wir in Lektion 6 bereits dargestellt, jedoch wollen wir diese wichtigen Funktionen hier noch einmal ausführlich schildern:

(1) Im Arbeitsspeicher steht folgendes Testprogramm:

```
10 REM **********
20 REM * UEBUNG *
30 REM **********
40 CLS
50 PRINT "TEST"
60 PRINT "VERLIEF"
70 PRINT "MIT"
80 PRINT "ERFOLG!"
90 END
```

(2) Um das Programm auf (Band-)Kassette[1] zu speichern, erteilen wir das Kommando SAVE "UEBUNG" : [2]

```
.
.
80 PRINT "ERFOLG!"
90 END
SAVE "UEBUNG"
```



(3) Der Rechner gibt daraufhin folgende Nachricht auf dem Bildschirm aus:

```
↓ RECORD.PLAY
```

Hierdurch erfährt der Anwender, daß die [RECORD]-Taste zu betätigen ist - die [PLAY]-Taste ist mit der [RECORD]-Taste gekoppelt und wird automatisch mitgedrückt.

Danach meldet das System Writing "UEBUNG"[3] und nach Beendigung des Vorgangs Ready:

```
Writing   "UEBUNG"
Ready
```



(4) Drücken Sie jetzt am Kassettengerät die Tasten [STOP/EJECT] und anschließend [REWIND] . Nach Abschluß des Rückspulvorgangs ist wieder [STOP/EJECT] zu betätigen.

(5) Mit dem Kommando VERIFY "UEBUNG"[4] wollen wir dann überprüfen, ob das Programm korrekt gespeichert wurde. Nachdem die Anweisung erteilt worden ist, erscheint ↓ PLAY auf dem Bildschirm. Wir drücken daher auf [PLAY]. Der Rechner reagiert im Regelfall mit den folgenden Nachrichten:

---

1) Es ist dringend zu empfehlen, hierfür nicht die Kassette zu verwenden, auf der sich der BASIC-Interpreter befindet.

2) Der in Anführungsstrichen angegebene Programmname ("UEBUNG") kann auch fortgelassen werden. Dann würde das System eine Speicherung ohne Programmnamen vornehmen.

3) frei übersetzt: (Das Programm) "UEBUNG" wird jetzt (auf Band) geschrieben.

4) von engl. "to verify" = verifizieren, auf Richtigkeit prüfen

```
VERIFY "UEBUNG"
⭳ PLAY
Found      "UEBUNG"
VERIFYING "UEBUNG"
OK
Ready
```

Nur im Fehlerfalle, der beispielsweise bei Verwendung mangelhafter Kassetten auftreten kann, gibt das System zum Schluß folgende Meldung aus:

```
 .
 .
READ error
Ready
```

In einer solchen Situation sollte man zunächst versuchen, die geschilderte Prozedur zu wiederholen.

(7) Nach der erfolgreichen "Überprüfung auf Richtigkeit" betätige man - in der vorgegebenen Reihenfolge - die Tasten [STOP/EJECT], [REWIND], [STOP/EJECT].

Nun können wir den Arbeitsspeicher mit dem Kommando NEW löschen, um uns anschließend davon zu überzeugen, daß das Laden des auf Kassette befindlichen Programms ordnungsgemäß funktioniert:

	Freie Übersetzungen:
.	
.	
NEW	Lösche den Arbeitsspeicher!
Ready	Fertig (Systemmeldung)
LOAD "UEBUNG"	Lade das Programm "UEBUNG"!
⭳ PLAY	Drücke auf die PLAY-Taste des Bandgeräts! (Systemmeldung)
Found "UEBUNG"	Das Programm "UEBUNG" wurde gefunden. (Systemmeldung)
LOADING "UEBUNG"	Jetzt wird das Programm "UEBUNG" geladen. (Systemmeldung)
Ready	Fertig (Systemmeldung)

PE 27.2

(1) Das im Arbeitsspeicher befindliche Programm soll mit dem Namen "TEST" auf Bandkassette gespeichert werden. Wie lautet das Systemkommando?

. . . .

(2) Auf einer Bandkassette wurde das Programm "TEST" gespeichert und anschließend zurückgespult. Wie lautet das Systemkommando, mit dem man das im Arbeitsspeicher befindliche Programm mit jenem auf Band ("TEST") vergleicht?

. . . .

(3) Die Bandkassette ist zurückgespult worden, und Sie sollen das Programm "TEST" in den Arbeitsspeicher laden. Wie lautet das Systemkommando?

. . . .

LÖS 27.2

(1) Das im Arbeitsspeicher befindliche Programm soll mit dem Namen "TEST" auf Bandkassette gespeichert werden. Das Systemkommando lautet

SAVE "TEST" .

(2) Auf einer Bandkassette wurde das Programm "TEST" gespeichert und anschließend zurückgespult. Das Systemkommando, mit dem man das im Arbeitsspeicher befindliche Programm mit jenem auf Band ("TEST") vergleicht, lautet

VERIFY "TEST" .

(3) Das Systemkommando zum Laden des Programms "TEST" lautet

LOAD "TEST" .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 27.2** !

LE 27.3

In der vorangegangenen **LE 27.2** haben wir gelernt, wie man Programme auf Bandkassette speichert oder von dort lädt und auch die Richtigkeit der Aufzeichnung überprüfen kann.

Eine weitere Möglichkeit der sinnvollen Nutzung eines Kassettengeräts besteht darin, umfangreiche Daten auf das Band auszugeben und bei Bedarf wieder in den Arbeitsspeicher zu lesen, um sie einer erneuten Bearbeitung zu unterziehen.

Mit dem folgenden Musterprogramm, daß man für individuelle Bedürfnisse leicht ändern kann, sollen die Adressen der Mitglieder eines Vereins auf Band geschrieben werden.

```
10 REM ********************
20 REM * ERSTELLUNG EINER *
30 REM *      DATEI       *
40 REM ********************
50 CLS
60 PRINT "AM BANDGERAET BITTE"
70 PRINT [7, 0] "RECORD/PLAY";
80 PRINT "-TASTE DRUECKEN!"
90 WOPEN "ADRESSEN"
100 CLS
110 PRINT
120 PRINT [0, 7] TAB(10) "* DATENERFASSU
NG *"
130 PRINT : PRINT
140 PRINT "NACHNAME"
150 PRINT : PRINT
160 PRINT "VORNAME"
170 PRINT : PRINT
180 PRINT "STRASSE/NR."
190 PRINT : PRINT
200 PRINT "WOHNORT"
```

```
210 CURSOR 12, 4
220 INPUT NA$
230 IF NA$ = "XYZ" THEN 430
240 CURSOR 12, 7
250 INPUT VN$
260 CURSOR 12, 10
270 INPUT ST$
280 CURSOR 12, 13
290 INPUT WO$
300 CURSOR 12, 23
310 PRINT [7, 2] "SIND DIE EINGABEN RICH
TIG?";
320 GET A$
330 IF A$ = "J" THEN 410
340 IF A$ = "N" THEN 360
350 GOTO 320
360 CLS
370 PRINT : PRINT
380 PRINT "EINGABE WIEDERHOLEN!"
390 FOR I = 1 TO 800 : NEXT I
400 GOTO 100
410 PRINT/T NA$, VN$, ST$, WO$
420 GOTO 100
430 CLOSE
440 CLS
450 PRINT [7, 0] "STOP/EJECT";
460 PRINT " DRUECKEN U. GGF. AUCH"
470 PRINT [7, 0] "REWIND";
480 PRINT " !"
490 END
```

Erläuterungen:

Zeilen 10 - 40: Diese Befehle dienen nur der Kennzeichnung des Programms.

Zeile 50: Hiermit löscht man den Bildschirm.

Zeilen 60 - 80: Sie verursachen folgende Nachricht auf dem Bildschirm:

```
AM BANDGERAET BITTE
RECORD/PLAY-TASTE DRUECKEN!
```

"RECORD/PLAY" erscheint mit weißen Buchstaben auf schwarzem Grund. (Achtung: Die Angaben nach PRINT in Zeile 70 (= Farbspezifikation) müssen in eckigen - nicht in runden Klammern - aufgeführt werden; vgl. hierzu Seite 16-17 ff., vor allem Seite 16-19!)

Diese Nachricht geben wir aus, weil der Rechner für die Ausführung des Befehls in

Zeile 90 erwartet, daß man RECORD/PLAY betätitgt - eine diesbezügliche Systemmeldung erhält man nämlich nicht.

Wozu dient die Anweisung in Zeile 90?

Bevor Elemente einer Datei auf Band geschrieben werden können, muß sie im Programm eröffnet worden sein. Hierfür verwenden wir

WOPEN "Programmname" (von engl. "open for writing" = eröffne für das Schreiben).

Durch diese Instruktion wird ein Ein-/Ausgabekanal "eingerichtet" und auf Band bestimmte Anfangsmarkierungen und der angegebene Programmname geschrieben.

Zeile 100: Löschung des Bildschirms

Zeilen 110 - 220 erzeugen das folgende Bildschirmbild:

```
        * DATENERASSUNG *

NACHNAME     ? ▒

VORNAME

STRASSE/NR.

WOHNORT
```

Dabei verursachten der Befehl der Zeile 210 "einen Rücksprung im aufgebauten Bild" und die Anweisung der Zeile 220 das Fragezeichen. In

Zeile 230 erfolgt die Abfrage nach "XYZ" (= Endekriterium). Im JA-Fall verzweigt das Programm zum Befehl der Zeile 430. Mit der Instruktion der

Zeilen 240 - 290 werden die gewünschten Eingaben vorgenommen, um dann mit den Anweisungen der

Zeilen 320 - 350 zu klären, ob die Eingaben richtig waren (dann Sprung nach Zeile 410) oder nicht (dann Ausgabe der Nachricht "EINGABE WIEDERHOLEN!" und im Anschluß an eine kurze Verzögerungsschleife in Zeile 390 erfolgt ein Rücksprung zur Zeile 100).

Mit Hilfe des Befehls der Zeile 210, nämlich

PRINT/T ... (von engl. "to print on tape, hier frei zu übersetzen mit "auf Band schreiben")

werden die angegebenen Variablen auf Band geschrieben. Allerdings bedeutet dies nicht, daß effektiv physikalisch schon auf Band ausgegeben wird. Das System speichert die auszugebenden Daten nämlich zunächst in einem Ausgabepuffer und erst wenn dieser gefüllt ist, erfolgt automatisch die Übertragung auf Band. Hierum braucht sich der Programmierer bzw. der Anwender allerdings nicht zu kümmern.

Mit der Anweisung

CLOSE (von engl. "to close" = schließen)

in Zeile 430 schließen wir die Banddatei, d. h. die letzten Daten im Ausgabepuffer werden auf Band geschrieben und bestimmte Endemarkierungen gesetzt. Gleichzeitig erfolgt die Freigabe des Ein-/Ausgabekanals. - Durch die Anweisungen der

Zeilen 450 - 490 werden die notwendigen "Abschlußarbeiten" veranlaßt.

PE 27.3

(Bitte, die linke Seite abdecken!)

(1) Es ist eine Banddatei, auf die Daten ausgegeben werden sollen, mit dem Namen "VEREIN" zu eröffnen. Wie lautet der Befehl?

. . . .

(2) Auf eine mit dem Namen "VEREIN" eröffnete Banddatei sollen die Variablen A1$, A2$ und A3$ ausgegeben werden. Wie lautet die Anweisung?

. . . .

(3) Nach Erkennung des Endekriteriums soll eine Datei geschlossen werden. Wie lautet der Befehl?

. . . .

LÖS 27.3

(1) Es ist eine Banddatei, auf die Daten ausgegeben werden sollen, mit dem Namen "VEREIN" zu eröffnen. Der Befehl lautet

.. WOPEN "VEREIN" .

(2) Auf eine mit dem Namen "VEREIN" eröffnete Banddatei sollen die Variablen A1$, A2$ und A3$ ausgegeben werden. Die Anweisung lautet

.. PRINT/T A1$, A2$, A3$.

(3) Nach Erkennung des Endekriteriums soll eine Datei geschlossen werden. Der Befehl lautet

.. CLOSE .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 27.3** !

LE 27.4

Wir unterstellen, daß mit dem in der vorangegangenen LE 27.3 erläuterten Programm eine Adressdatei erstellt worden ist. Mit dem folgenden Musterprogramm, daß man wiederum für individuelle Bedürfnisse leicht ändern kann, sollen die Daten vom Band in den Arbeitsspeicher gelesen und dann auf dem Plotter-Drucker ausgegeben werden.

```
10 REM ********************
20 REM * LESEN EINER DATEI *
30 REM ********************
40 CLS
50 PRINT "AM BANDGERAET BITTE"
60 PRINT [7, 0] "PLAY";
70 PRINT "-TASTE DRUECKEN!"
80 ON ERROR GOTO 1000
90 ROPEN "ADRESSEN"
100 PRINT/P "MITGLIEDER DES VEREINS:"
110 PRINT/P "-----------------------"
120 INPUT/T NA$, VN$, ST$, WO$
130 PRINT/P NA$; ", ";
140 PRINT/P VN$
150 PRINT/P ST$
160 PRINT/P WO$
170 PRINT/P
180 GOTO 120
190 CLOSE
200 CLS
210 PRINT [7, 0] "STOP/EJECT";
220 PRINT " DRUECKEN U. GGF. AUCH"
230 PRINT [7, 0] "REWIND";
240 PRINT "!"
250 END
1000 REM FEHLERBEHANDLUNGSROUTINE
1010 REM -----------------------
1020 IF (ERL = 120) * (ERN = 63) THEN RE
SUME 190
1030 ON ERROR GOTO 0
```

Programmliste:

```
10 REM ********************
20 REM * LESEN EINER DATEI *
30 REM ********************
40 CLS
50 PRINT "AM BANDGERAET BITTE"
60 PRINT [7, 0] "PLAY";
70 PRINT "-TASTE DRUECKEN!"
80 ON ERROR GOTO 1000
90 ROPEN "ADRESSEN"
100 PRINT/P "MITGLIEDER DES VEREINS:"
110 PRINT/P "----------------------"
120 INPUT/T NA$, VN$, ST$, WO$
130 PRINT/P NA$; ", ";
140 PRINT/P VN$
150 PRINT/P ST$
160 PRINT/P WO$
170 PRINT/P
180 GOTO 120
190 CLOSE
200 CLS
210 PRINT [7, 0] "STOP/EJECT";
220 PRINT " DRUECKEN U. GGF. AUCH"
230 PRINT [7, 0] "REWIND";
240 PRINT "!"
250 END
1000 REM FEHLERBEHANDLUNGSROUTINE
1010 REM ------------------------
1020 IF (ERL = 120) * (ERN = 63) THEN RE
SUME 190
1030 ON ERROR GOTO 0
```

Erläuterungen:

Zeilen 10 - 30: Diese Befehle dienen nur der Kennzeichnung des Programms.

Zeile 40: Hiermit löscht man den Bildschirm.

Zeilen 50 - 70: Sie verursachen folgende Nachricht auf dem Bildschirm:

```
AM BANDGERAET BITTE
PLAY-TASTE DRUECKEN!
```

"PLAY" erscheint mit weißen Buchstaben auf schwarzem Grund. (Achtung: Die Angaben nach PRINT in Zeile 60 (= Farbspezifikation) müssen in eckigen - nicht in runden Klammern - aufgeführt werden; vgl. hierzu Seite 16-17 ff., vor allem Seite 16-19!)

Diese Nachricht geben wir aus, weil der Rechner für die Ausführung des Befehls in

Zeile 90 erwartet, daß man [PLAY] betätigt - eine diesbezügliche Systemmeldung erhält man nämlich nicht.

(Die Anweisungen der Zeilen 80 und 1000 - 1030 werden wir auf Seite 27-21 näher behandeln.)

Bevor Elemente einer Datei von Band gelesen werden können, muß sie im Programm eröffnet worden sein. Hierfür verwenden wir

ROPEN "Programmname" (von engl. "open for reading" = eröffne für das Lesen).

Durch diese Instruktion wird ein Ein-/Ausgabekanal "eingerichtet" und die Datei mit dem angegebenen Programmnamen auf Band gesucht.

Zeilen 100 - 110: Ausgabe einer Überschrift auf dem Plotter-Drucker

Programmliste:

```
10 REM ********************
20 REM * LESEN EINER DATEI *
30 REM ********************
40 CLS
50 PRINT "AM BANDGERAET BITTE"
60 PRINT [7, 0] "PLAY";
70 PRINT "-TASTE DRUECKEN!"
80 ON ERROR GOTO 1000
90 ROPEN "ADRESSEN"
100 PRINT/P "MITGLIEDER DES VEREINS:"
110 PRINT/P "-----------------------"
120 INPUT/T NA$, VN$, ST$, WO$
130 PRINT/P NA$; ", ";
140 PRINT/P VN$
150 PRINT/P ST$
160 PRINT/P WO$
170 PRINT/P
180 GOTO 120
190 CLOSE
200 CLS
210 PRINT [7, 0] "STOP/EJECT";
220 PRINT " DRUECKEN U. GGF. AUCH"
230 PRINT [7, 0] "REWIND";
240 PRINT "!"
250 END
1000 REM FEHLERBEHANDLUNGSROUTINE
1010 REM -----------------------
1020 IF [ERL = 120] * [ERN = 63] THEN RE
SUME 190
1030 ON ERROR GOTO 0
```

Zeile 120: Mit Hilfe der Anweisung

INPUT/T Variable1, ... VariableN (von engl. "input from tape = Eingabe von Band)

lesen wir Daten vom Band in den Arbeitsspeicher, die der Plotter-Drucker dann aufgrund der

Zeilen 130 - 170 ausgibt.

Wegen des Sprungbefehls in

Zeile 180 wiederholt das System die Instruktionen der Zeilen 120 - 170 so lange, bis alle Daten der Datei gelesen und verarbeitet worden sind. Wenn dann die Anweisung in Zeile 120 nochmals ausgeführt werden soll, würde normalerweise die Verarbeitung mit folgender Fehlermeldung unterbrochen:

```
Out of file error in 120
Ready
```

frei übersetzt: Fehler wegen Dateiendes in Zeile 120

Durch ON ERROR GOTO 1000 in

Zeile 80 haben wir aber sichergestellt, daß im Fehlerfalle die Verarbeitung mit der Fehlerbehandlungsroutine (ab Zeile 1000) fortgesetzt wird. In

Zeile 1020 untersuchen wir, ob der erwartete Fehler - nämlich ein "Out of file error" (ERN = 63) in Zeile 120 (ERL = 120) - aufgetreten ist, um dann die Verarbeitung in Zeile 190 wieder aufzunehmen; andernfalls erfolgt durch die Anweisung in Zeile 1030 die Ausgabe einer Fehlermeldung (vgl. hierzu Seite 22-31 ff., vor allem Seite 22-37 f., Pkt. (3)!).

Zeile 190: Mit CLOSE schließen wir die Banddatei und geben den Ein-/Ausgabekanal wieder frei.

Durch die Anweisungen der

Zeilen 210 - 250 werden die notwendigen "Abschlußarbeiten" veranlaßt.

PE 27.4

(1) Es ist eine Banddatei mit dem Namen "FAKTEN", von der Daten gelesen werden sollen, zu eröffnen. Wie lautet der Befehl?

. . . .

(2) Von einer für Lesezwecke eröffneten Datei sollen Daten in die Hauptspeicherfelder A$, A, B$, B gelesen werden. Wie lautet der Befehl?

. . . .

(3) Eine Banddatei, von der gelesen wurde, ist zu schließen. Wie lautet der Befehl?

. . . .

LÖS 27.4

(1) Es ist eine Banddatei mit dem Namen "FAKTEN", von der Daten gelesen werden sollen, zu eröffnen. Der Befehl lautet

.. ROPEN "FAKTEN" .

(2) Von einer für Lesezwecke eröffneten Datei sollen Daten in die Hauptspeicherfelder A$, A, B$, B gelesen werden. Der Befehl lautet

.. INPUT/T A$, A, B$, B .

(3) Eine Banddatei, von der gelesen wurde, ist zu schließen. Der Befehl lautet

.. CLOSE .

Sollten Ihre Lösungen nicht richtig sein, kehren Sie zurück zur → **LE 27.4** !

ÜBUNGsprogramm zu Lektion 27

Wir sind hiermit am Ende des Buches angekommen, und der Verfasser möchte nicht versäumen, dem Leser für seine Geduld und Ausdauer zu danken und für das weitere Programmieren mit dem SHARP MZ-700 unendliches Vergnügen zu wünschen. Dieser Wunsch ist im normalen Sprachgebrauch eher als Leerformel anzusehen - in BASIC können wir ihn aber mit unserem letzten **ÜBUNG**sprogramm wirklich zum Ausdruck bringen:

Programmliste:

```
10 PRINT "VIEL VERGNUEGEN!"
20 GOTO 10
```

Für Ihre weitere Betätigung mit dem SHARP MZ-700 soll Ihnen der umfangreiche Anhang ("Systemkommandos", "Steuercodes", "ASCII-Code-Tabelle", "Code-Tabelle für den Bildschirmzeichenspeicher", "ASCII-Code-Tabelle für den Plotter-Drucker", "Systemfehlermeldungen", "Abkürzungen für BASIC-Schlüsselwörter", "Sprachliche Erläuterungen zu den Schlüsselwörtern", "Erstellung einer Sicherungskopie", "Literaturhinweise", "Stichwortverzeichnis") eine Hilfe sein. Außerdem verweisen wir auf die Liste der vordefinierten Funktionen (Seite 24-12).

Darüber hinaus machen wir darauf aufmerksam, daß für den SHARP MZ-700 umfangreiches Zubehör und anschließbare Peripherie existieren, wodurch sich neue Anwendungsbereiche erschließen lassen. Die programmtechnische Behandlung der Geräte können Sie den beigefügten Gebrauchsanweisungen des Herstellers entnehmen, die Sie mit dem durch dieses Buch vermittelten Grundwissen ohne Schwierigkeiten verstehen werden.

ANHANG

Systemkommandos

Bei Programmiersprachen, die für "Interpreter-Betrieb" konzipiert wurden, ist die Unterteilung von Anweisungen in Programmbefehle einerseits und Systemkommandos andererseits nicht ohne Willkür, da hier letztlich nahezu jede Anweisung sowohl als Programmbefehl (mit Zeilennummer) als auch als Systemkommando (ohne Zeilennummer) verwendet werden kann. Die Zuordnung haben wir daher nach der üblichen Verwendungsart der Anweisung vorgenommen.

Die derart als Systemkommando klassifizierten Anweisungen sind in folgender Reihenfolge beschrieben:

Format: Hier wird das korrekte Format (s. u.) des Systemkommandos aufgeführt.

Funktion: Unter diesem Punkt steht immer ein kurzer Hinweis, für welchen Zweck das Systemkommando eingesetzt werden kann.

Bemerkungen: An dieser Stelle erfährt man Details und Besonderheiten zum Systemkommando, u. U. auch mehrere Anwendungsbeispiele.

Erläuterungen zu den Formatangaben:

(1) Die mit Großbuchstaben aufgeführten Elemente des Formats sind reservierte BASIC-Wörter und müssen ggf. in der angegebenen Weise verwendet werden.

Beispiel: LOAD ["Programmname"]

(2) Angaben in eckigen Klammern können je nach Wunsch des Programmierers eingefügt oder weggelassen werden.

Beispiel: LOAD ["Programmname"]

(3) Die innerhalb von runden Klammern mit Kleinbuchstaben aufgeführten Elemente des Formats sind vom Benutzer anzugeben.

Beispiel: DELETE (Zeilennummer - Zeilennummer)

Systemkommandos:

AUTO

Format:

AUTO [Zeilennummer [, Schrittweite]]

Funktion:

Das Kommando erzeugt automatisch eine Zeilennummer nach jedem Drücken der CR-Taste.

Bemerkungen:

AUTO mit den wahlfreien Angaben: Die Zeilennumerierung beginnt mit der angegebenen Zeilennummer; die nach dem Drücken der CR-Taste folgende Zeilennummer ist um die angegebene Schrittweite höher als die vorangehende Zeilennummer.

Beispiele:

AUTO 30, 3
erzeugt die Zeilennummern 30, 33, 36, 39, 42, ... ;

AUTO 100, 20
erzeugt die Zeilennummern 100, 120, 140, 160, 180, ...

Wenn AUTO ohne die wahlfreien Zusätze verwendet wird, dann beginnt die Zeilennumerierung mit 10, und die nach dem Drücken der CR-Taste folgende Zeilennummer ist um die Schrittweite 10 höher als die vorangehende Zeilennummer.

Beispiel:

AUTO
erzeugt die Zeilennummern 10, 20, 30, 40, 50, ...

Wenn das Systemkommando AUTO eine Zeilennummer erzeugt, die schon mit einem Befehl "belegt" ist, dann wird zusätzlich die Anweisung aufgelistet, um den Benutzer zu warnen, daß eine Befehlseingabe unter dieser Zeilennummer einen schon existierenden Befehl überschreiben würde. Druckt man jedoch unmittelbar nach dem Erscheinen der mit einer Anweisung belegten Zeilennummer die CR-Taste, dann bleibt der alte Befehl erhalten, und die nächste Zeilennummer erscheint.

Man beendet die automatische Zeilennumerierung durch gleichzeitige Betätigung von SHIFT und BREAK.

Systemkommandos:

BYE Format:

BYE

Funktion:

Dieses Kommando übergibt die Steuerung des Systems vom BASIC-Interpreter an das BASIC-eigene Monitorprogramm im RAM-Speicherbereich.

(Einzelheiten zu den Monitor-Befehlen entnehmen Sie bitte den Systemunterlagen des Herstellers!)

CLR Format:

CLR

Funktion:

Das Kommando löscht alle Variablen und Feldvereinbarungen. Numerischen Variablen wird eine 0 und String-Variablen ein Nullstring (" ") zugewiesen. Ebenfalls gelöscht werden alle mittels DEF FN-Anweisungen getroffenen Funktionsvereinbarungen.

CONSOLE Formate:

CONSOLE
CONSOLE (ZeileX, Zeilenzahl)
CONSOLE (ZeileX, Zeilenzahl, SpalteX, Spaltenzahl)

Funktion:

Mit diesem Kommando bestimmt man den sog. Scroll-Bereich (von engl. "scroll" = Liste, Rolle) des Bildschirms.

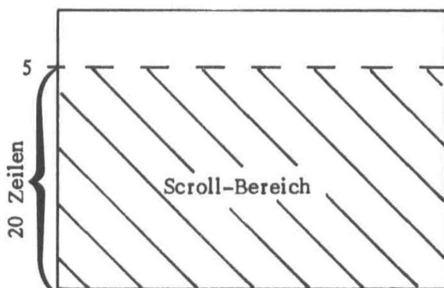
Normalerweise ist der ganze Bildschirm als Scroll-Bereich (= Schreib-Bereich) vorgesehen. Er läßt sich jedoch verkleinern, so daß nur noch im eingeengten Feld Verarbeitungsergebnisse wiedergegeben werden können. Auch wird nur die (ggf. eingeengte) Scroll-Fläche durch CLS gelöscht.

Beispiele:

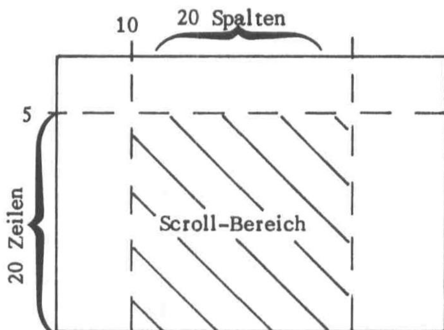
CONSOLE 5, 20

Systemkommandos:

definiert den Bildschirm ab Zeile 5 (dann 20 Zeilen, d. h. den verbleibenden Rest) als Scroll-Bereich:



CONSOLE 5, 20, 10, 20
definiert den Bildschirm ab Zeile 5 (dann 20 Zeilen, d. h. bis zum unteren Rand) und ab Spalte 10 für 20 Spalten als Scroll-Bereich:



CONSOLE
hebt vorangegangene Einschränkungen wieder auf.

Achtung: Die Einengung des Scroll-Bereichs erschwert auch die Möglichkeiten zur Korrektur des Programms.

Systemkommandos:

CONT

Format:

CONT

Funktion:

Das Kommando bewirkt die Fortsetzung der Programmausführung, die durch Betätigung der Tasten [SHIFT] und [BREAK] oder durch einen STOP- oder END-Befehl unterbrochen wurde.

DEF KEY

Format:

DEF KEY ((NR) = "String")

Funktion:

Mittels DEF KEY ... können die Funktionstasten (die großen blauen Tasten [F1] bis [F5]) insgesamt mit bis zu 10 unterschiedlichen Strings belegt werden. Dies macht es dann möglich, daß man durch das Drücken nur einer Taste nicht nur ein einziges Zeichen, sondern gleich eine ganze Zeichenkette (String) eingeben kann. Die Betätigung einer der Tasten [F1] bis [F5] entspricht den Funktionstasten-Nummern 1 bis 5, das gleichzeitige Drücken der [SHIFT]-Taste und einer der Tasten [F1] bis [F5] den Funktionstasten-Nummern 6 bis 10. In dem dargestellten Anweisungsformat entspricht NR der Funktionstasten-Nummer und "String" der Zeichenkette, die der durch NR spezifizierten Funktionstaste zugeordnet werden soll. Ein DEF KEY-Kommando löscht frühere Belegungen.

Beispiele:

DEF KEY (1) = "GOSUB"
bewirkt, daß bei Betätigung der Funktionstaste [F1] das Befehlswort GOSUB auf dem Bildschirm erscheint.

DEF KEY (2) = "LIST" + CHR$(13)
bewirkt, daß bei Betätigung der Funktionstaste [F2] LIST auf dem Bildschirm erscheint; zusätzlich wird das LIST-Kommando unverzüglich ausgeführt, denn CHR$(13) entspricht einer Betätigung der [CR]-Taste.

(Vgl. in diesem Zusammenhang auch das Systemkommando KEY LIST !)

Systemkommandos:

DELETE Formate:

DELETE (Zeilennummer)
DELETE (- Zeilennummer)
DELETE (Zeilennummer -)
DELETE (Zeilennummer - Zeilennummer)

Funktion:

Das Kommando dient zum Löschen von Programmzeilen.

Bemerkungen:

Ohne Angabe einer Zeilennummer meldet sich das System mit der Fehlernachricht "Syntax error".

Beispiele:

DELETE 50
löscht Programmzeile 50 .

DELETE - 50
löscht alle Programmzeilen bis einschließlich 50 .

DELETE 50 -
löscht das Programm ab Zeile 50 .

DELETE 50 - 200
löscht alle Programmzeilen von 50 bis einschließlich 200 .

KEY LIST Format:

KEY LIST

Funktion:

Dieses Kommando gibt eine Liste der Strings aus, die den frei definierbaren Funktionstasten zugeordnet sind.

Beispiel:

```
KEY LIST
DEF KEY(1)="RUN"+CHR$(13)
DEF KEY(2)="LIST"
DEF KEY(3)="AUTO"
DEF KEY(4)="RENUM"
```

S y s t e m k o m m a n d o s :

DEF KEY(5)="COLOR"
DEF KEY(6)="CHR$("
DEF KEY(7)="DEF KEY("
DEF KEY(8)="CONT"
DEF KEY(9)="SAVE"
DEF KEY(10)="LOAD"

LIST

Formate:

LIST
LIST (Zeilennummer)
LIST (Zeilennummer -)
LIST (- Zeilennummer)
LIST (Zeilennummer - Zeilennummer)

Funktion:

Das Kommando bewirkt das Auflisten des im Arbeitsspeicher befindlichen Programms (oder von Teilen davon) am Bildschirm.

Bemerkungen:

Falls die Programmliste größer ist als eine Bildschirmseite, ergibt sich ein "Scrolling-Effekt", d. h. wenn am unteren Bildschirmrand eine neue Zeile erscheint, verschwindet jene am oberen Bildschirmrand. Das "Scrolling" kann man vorübergehend durch Niederdrücken der Leertaste unterbrechen; das Auflisten wird fortgesetzt, wenn man die Leertaste nicht mehr betätigt. Die Ausgabe kann durch gleichzeitiges Drücken von SHIFT und BREAK vorzeitig beendet werden.

B e i s p i e l e :

LIST
bewirkt die Auflistung des gesamten Programms.

LIST 30
bewirkt die Auflistung (nur) der Befehlszeile 30 .

LIST 30 -
bewirkt die Auflistung des Programmteils ab Zeile 30 .

LIST - 30
bewirkt die Auflistung des Programmteils bis Zeile 30 .

Systemkommandos:

LIST 30 - 90
bewirkt die Auflistung des Programmteils von Zeile 30 bis Zeile 90 .

LIST/P

Formate:

LIST/P
LIST/P (Zeilennummer)
LIST/P (Zeilennummer -)
LIST/P (- Zeilennummer)
LIST/P (Zeilennummer - Zeilennummer)

Funktion:

Das Kommando bewirkt das Auflisten des im Arbeitsspeicher befindlichen Programms (oder von Teilen davon) auf dem Plotter-Drucker.

Bemerkungen:

Die beim Systemkommando LIST aufgeführten Beispiele gelten in analoger Weise.

LOAD

Format:

LOAD ["Programmname"]

Funktion:

Das Kommando dient zum Laden eines BASIC-Programms oder eines Programms in Maschinensprache, das mit einem BASIC-Programm verkettet werden soll, von der Kassette in den Arbeitsspeicher.

Bemerkungen:

Beim Laden eines BASIC-Programms schließt das LOAD-Kommando automatisch alle offenen Dateien und löscht alle im Arbeitsspeicher befindlichen Daten und Programme (Programmteile), bevor es das (angegebene) Programm in den Arbeitsspeicher überträgt.

Beispiele:

LOAD
lädt das nächste BASIC-Programm von Kassette.

Systemkommandos:

LOAD "HEINO"
sucht auf der Kassette das Programm mit dem Namen "HEINO" und lädt es nach dem Auffinden in den Arbeitsspeicher.

Lädt man mittels des LOAD-Kommandos aus einem BASIC-Programm ein Maschinenprogramm, das mit dem BASIC-Programm verbunden werden soll, dann muß zur Reservierung eines Speicherbereichs für das Maschinenprogramm zuvor eine LIMIT-Anweisung ausgeführt worden sein. Das Maschinenprogramm wird unmittelbar nach dem Ladevorgang gestartet, wenn sich die Adresse innerhalb des mit LIMIT reservierten Bereichs befindet.

Beispiel:

```
.
.
200 LIMIT $BFFF
210 LOAD "BUDELMANN"
220 USR($C000)
.
.
.
.
```

(Vgl. hierzu die Systemunterlagen des Herstellers!)

MERGE Format:

MERGE ["Programmname"]

Funktion:

Mit diesem Programm verbindet man im Arbeitsspeicher das dort gespeicherte Programm mit einem zweiten, das auf Kassette gespeichert wurde. Fehlt der Programmname bei der Eingabe des MERGE-Kommandos, dann liest das System das nächste auf Kassette befindliche Programm.

Wenn die beiden zu verbindenden Programme Befehle mit gleichen Zeilennummern haben, dann werden die Befehlszeilen im Arbeitsspeicher durch jene mit gleicher Nummer des "externen" Programms überschrieben.

Nach Ausführung des Kommandos muß das Programm ggf. mit RUN gestartet werden.

Systemkommandos:

Beispiel:

MERGE "AUSWERTUNG"

MODE GR Format:

MODE GR

Funktion:

Das Kommando schaltet den Plotter-Drucker in den Grafik-Modus. Anschließend kann man die für diese Betriebsart vorgesehenen Befehle nutzen.

MODE TL Format:

MODE TL

Funktion:

Das Kommando schaltet den Plotter-Drucker in den Text-Modus mit 26 Zeichen pro Zeile.

MODE TN Format:

MODE TN

Funktion:

Das Kommando schaltet den Plotter-Drucker in den Text-Modus mit 40 Zeichen pro Zeile.

MODE TS Format:

MODE TS

Funktion:

Das Kommando schaltet den Plotter-Drucker in den Text-Modus mit 80 Zeichen pro Zeile.

Systemkommandos:

NEW

Format:

NEW

Funktion:

Das Kommando löscht das im Arbeitsspeicher befindliche BASIC-Programm incl. der Datenfelder, nicht hingegen den Speicherbereich für Maschinenprogramme, der durch die LIMIT-Anweisung eingerichtet wurde.

Wenn man mit dem Rechner gearbeitet hat und ein neues Programm schreiben will, sollte vorher grundsätzlich N E W erteilt werden. Vor einem LOAD-Kommando ist es hingegen nicht erforderlich, mit N E W den Arbeitsspeicher zu löschen, da L O A D diese Funktion zusätzlich übernimmt.

PAGE

Format:

PAGE (Zeilenzahl)

Funktion:

Dieses Kommando legt die Zahl der Zeilen fest, die ohne Zwischenraum auf dem Plotter-Drucker auszugeben sind, um dann automatisch eine Leerzeile hinzuzufügen. Anders ausgedrückt: Die auf dem Endlospapier gedruckten Ergebnisse werden automatisch durch eine Leerzeile getrennt, wenn die spezifizierten Zeilen geschrieben worden sind, weil dadurch für Ablagezwecke ein einfacheres Zerschneiden der Papierbahn möglich ist.

Der nach P A G E anzugebende Wert (Konstante oder Variable) darf zwischen 1 und 72 liegen.

PCOLOR

Format:

PCOLOR (Farbnummer)

Funktion:

Dieses Kommando bestimmt die Druckfarbe des Plotter-Druckers, wobei für die Farbnummer ein Wert von 0 - 3 anzugeben ist, und zwar entspricht 0 = schwarz, 1 = blau, 2 = grün, 3 = rot.

Systemkommandos:

Beispiel:

PCOLOR 2
stellt den Plotter-Drucker auf grüne Druckfarbe.

PLOT OFF Format:

PLOT OFF

Funktion:

Dieses Kommando hebt die Wirkung von PLOT ON wieder auf. (Vgl. hierzu PLOT ON !)

PLOT ON Format:

PLOT ON

Funktion:

Dieses Kommando ermöglicht die Verwendung des Plotter-Druckers als Anzeigeeinheit, wenn kein Bildschirm angeschlossen ist. Damit kann der SHARP MZ-700 auch ohne Bildschirm betrieben werden. Diese Betriebsart ist naturgemäß nur mit dem eingebauten Plotter-Drucker im Text-Modus (MODE TN) möglich.

Bei nicht druckbaren Zeichen wird im Plot-Betrieb ein Punkt (.) gedruckt (vgl. hierzu die ASCII-Tabelle für den Plotter-Drucker auf Seite Anh.-29 - Anh.-32). Die Tasten [INST], [DEL] und [←] sind in dieser Betriebsart gesperrt. [CTRL] + [G] kann zum Wechsel der Druckfarbe verwendet werden.

RENUM Formate:

RENUM
RENUM (neue Zeilennummer)
RENUM (neue Zeilennummer),, (Schrittweite)
RENUM (neue Z.nummer), (alte Z.nummer), (Schrittw.)

Funktion:

Das Kommando dient zur Neunumerierung eines im Arbeitsspeicher befindlichen Programms.

Systemkommandos:

Bemerkungen:

RENUM
bewirkt eine Neunumerierung des gesamten Programms. Die erste neunumerierte Zeilennummer beginnt mit 10, die folgenden Zeilennummern sind um die Schrittweite 10 erhöht.

Beispiel:

(alte Zeilennumerierung:)
```
7 PRINT A$
14 PRINT B$
16 PRINT C$
```

(neue Zeilennumerierung nach RENUM:)
```
10 PRINT A$
20 PRINT B$
30 PRINT C$
```

RENUM (neue Zeilennummer)
ermöglicht die Neunumerierung des gesamten Programms, wobei festgelegt wird, mit welcher Zeilennummer die Neunumerierung beginnen soll. Die folgenden Zeilen sind um die Schrittweite 10 erhöht.

Beispiel:

(alte Zeilennumerierung:)
```
7 PRINT A$
14 PRINT B$
16 PRINT C$
```

(neue Zeilennumerierung nach RENUM 100:)
```
100 PRINT A$
110 PRINT B$
120 PRINT C$
```

RENUM (neue Zeilennummer),,(Schrittweite)
ermöglicht die Neunumerierung des gesamten Programms, wobei festgelegt wird, ab welcher Zeilennummer und mit welcher Schrittweite die Neunumerierung erfolgen soll.

Beispiel:

(alte Zeilennumerierung:)
```
7 PRINT A$
14 PRINT B$
16 PRINT C$
```

S y s t e m k o m m a n d o s :

(neue Zeilennumerierung nach RENUM 200,,50:)

```
200 PRINT A$
250 PRINT B$
300 PRINT C$
```

RENUM (neue Z.nummer), (alte Z.nummer), (Schrittw.)
ermöglicht die Neunumerierung eines Programm t e i l s, wobei festgelegt wird, ab welcher Zeile des ("alten") Programms (= alte Zeilennummer) die Neunumerierung zu beginnen hat. Mit der Spezifizierung von (neue Zeilennummer) und (Schrittweite) legt man fest, mit welcher Zeilennummer und mit welcher Schrittweite die Neunumerierung des Programmteils erfolgen soll.

B e i s p i e l :

(alte Zeilennumerierung:)

```
10 PRINT A$
20 PRINT B$
30 PRINT C$
40 PRINT A
50 PRINT B
60 PRINT C
```

(neue Zeilennumerierung nach RENUM 100, 40, 20:)

```
10 PRINT A$
20 PRINT B$
30 PRINT C$
100 PRINT A
120 PRINT B
140 PRINT C
```

Das Kommando R E N U M ändert auch die Sprungadressen der Befehle G O T O .., G O S U B .., .. T H E N .., O N G O T O .., O N G O S U B .. entsprechend der Neunumerierung. Aber: R E N U M kann nicht dazu verwendet werden, um die vorgegebene Reihenfolge der Befehle zu verändern. Außerdem: Bei der Spezifizierung von (neue Zeilennummer) und (alte Zeilennummer) ist darauf zu achten, daß der erste Wert größer ist als der zweite. Die Fehlermeldung "Illegal data error" wird ausgegeben, wenn durch R E N U M eine Zeilennummer erzeugt würde, die größer ist als 65 535.

Systemkommandos:

RUN

Format:

RUN [Zeilennummer]

Funktion:

Mit dem Kommando bewirkt man die Übersetzung und Ausführung eines BASIC-Programms. Alte Variableninhalte werden vorher gelöscht.

Bemerkungen:

RUN
übersetzt das im Arbeitsspeicher befindliche BASIC-Programm und führt es aus.

RUN (Zeilennummer)
bewirkt die Übersetzung und Ausführung (eines Teils) des im Arbeitsspeicher befindlichen BASIC-Programms, beginnend mit der angegebenen Zeilennummer.

SAVE

Format:

SAVE ["Programmname"]

Funktion:

Das Kommando dient zum Speichern eines im Arbeitsspeicher befindlichen BASIC-Programms auf Bandkassette.

Bemerkungen:

Mit diesem Kommando wird nur das im Arbeitsspeicher befindliche BASIC-Programm auf Kassette übertragen, nicht hingegen ein Maschinenprogramm, das im Speicherbereich für Maschinenprogramme gespeichert ist.

Beispiele:

SAVE "JUTTA"
bewirkt das Speichern des BASIC-Programms mit dem Namen "JUTTA". (Der Programmname darf maximal aus 16 Zeichen bestehen.)

SAVE
bewirkt das Speichern des BASIC-Programms ohne Programmnamen.

Systemkommandos:

SKIP

Format:

SKIP (Zeilenzahl)

Funktion:

Mit SKIP kann das Papier des Plotter-Druckers bis maximal 20 Zeilen vorwärts oder rückwärts transportiert werden. Die Anzahl der Zeilen ergibt sich aus dem nach dem Befehlswort SKIP anzugebenden Wert (Konstante oder Variable), der zwischen -20 und +20 liegen muß. Das Kommando darf nur im Text-Modus verwandt werden.

TEST

Format:

TEST

Funktion:

Dieses Kommando dient der Überprüfung der Funktionstüchtigkeit der vier Farbminen des Plotter-Druckers durch Ausgabe von vier Quadraten in den Farben Schwarz, Blau, Grün und Rot.

Wenn die Minen einen längeren Zeitraum nicht genutzt wurden, kann man durch mehrmalige Wiederholung des Kommandos u. U. eine "Aktivierung des Farbflusses" bewirken.

TROFF

Format:

TROFF

Funktion:

TROFF beendet die Ablaufverfolgung, die mit TRON einzuleiten ist. (Vgl. hierzu auch TRON !)

TRON

Format:

TRON

Funktion:

Das Kommando TRON (= "Trace on") dient zur Ablaufverfolgung des Programms. Die Zeilennummer des

Systemkommandos:

gerade ausgeführten Befehls wird am Bildschirm aufgelistet und bietet damit dem Anwender eine wertvolle Hilfe beim Auffinden von logischen Fehlern.

VERIFY

Format:

VERIFY "Programmname"

Funktion:

Das Kommando bewirkt die Überprüfung, ob ein auf Kassette gespeichertes Programm mit dem aktuellen, im Arbeitsspeicher befindlichen Programm übereinstimmt. Hierdurch können fehlerhafte Aufzeichnungen aufgrund von mangelhaftem Bandmaterial, Funktionsstörungen etc. erkannt werden.

Bemerkungen:

Beispiele:

VERIFY
überprüft das nächste Programm auf dem Kassettenband.

VERIFY "MAJA"
sucht auf der Kassette ein Programm mit dem Namen "MAJA" und vergleicht es nach dem Auffinden mit jenem im Arbeitsspeicher.

Anwendungsbeispiel:

Ein Programm "ALARICH" wurde mit SAVE gespeichert und das Band anschließend zurückgespult. Wenn man dann VERIFY mit der Tastatur geschrieben hat und die [CR]-Taste gedrückt worden ist, antwortet das System mit der Aufforderung ↓PLAY (= Drücke die [PLAY]-Taste auf dem Bandgerät!)

Nach Betätigung der [PLAY]-Taste meldet der Rechner schließlich

Found "ALARICH"

VERIFYING "ALARICH"

und schließlich

Systemkommandos:

OK

Ready,

d. h. die verglichenen Programme stimmen überein oder aber

READ error

Ready,

d. h. das Programm ist nicht richtig abgespeichert, und das SAVE-Kommando ist zu wiederholen.

Steuercodes

Steuercodes entsprechen in ihrer Wirkung den Systemkommandos. Man erteilt dem Rechner ein Kommando mittels Steuercode durch gleichzeitiges Drücken der [CTRL]-Taste und einer bestimmten Buchstabentaste.

Die nachfolgende Tabelle enthält neben den Steuercodes und den zugeordneten Funktionen die korrespondierenden ASCII-Werte. Steuercodes, die auch mittels der PRINT-Anweisung innerhalb eines BASIC-Programms verwendet werden können, sind mit einem Stern (*) gekennzeichnet; beispielsweise hat die gleichzeitige Betätigung von [CTRL] + [V] die gleiche Wirkung wie .. PRINT CHR$(22) bzw. wie CLS.

CTRL +	ASCII-Wert	Funktion
E	5	schaltet auf Kleinschreibung als Eingabemodus für alphanumerische Zeichen.
F	6	schaltet auf Großschreibung als Eingabemodus für alphanumerische Zeichen.
M	13 *	"Wagenrücklauf" ([CR])
P	16 *	Funktion wie [DEL]-Taste
Q	17 *	bewegt Cursor um eine Zeile nach unten.
R	18 *	bewegt Cursor um eine Zeile nach oben.
S	19 *	bewegt Cursor um eine Spalte nach rechts.
T	20 *	bewegt Cursor um eine Spalte nach links.
U	21 *	bewegt Cursor zur [HOME]-Position.
V	22 *	löscht den Bildschirm zur Hintergrundfarbe.
W	23 *	schaltet den Eingabemodus für Graphikzeichen ein ([GRAPH]).
X	24 *	fügt ein Leerzeichen ein ([INST]).
Y	25 *	schaltet den Eingabemodus für alphanumerische Zeichen ein.

Code-Tabelle für den Bildschirmzeichenspeicher

(Der Bildschirmzeichenspeicher beginnt ab Adresse 53 248 (= $D000).

SP	0	S	19	6	38		57		76		95		114
A	1	T	20	7	39		58		77	π	96		115
B	2	U	21	8	40		59		78	!	97		116
C	3	V	22	9	41		60	:	79	"	98		117
D	4	W	23	–	42		61	↑	80	#	99		118
E	5	X	24	=	43		62	<	81	$	100		119
F	6	Y	25	;	44		63	[	82	%	101		120
G	7	Z	26	/	45		64	♥	83	&	102		121
H	8	£	27	.	46	♠	65	]	84	'	103		122
I	9		28	,	47		66	@	85	(	104		123
J	10		29		48		67		86	)	105		124
K	11		30		49	♦	68	>	87	+	106		125
L	12		31		50	←	69	↓	88	*	107		126
M	13	0	32		51	♣	70		89		108		127
N	14	1	33		52	●	71	→	90		109		128
O	15	2	34		53	○	72		91		110	a	129
P	16	3	35		54	?	73		92		111	b	130
Q	17	4	36		55		74		93		112	c	131
R	18	5	37		56		75		94		113	d	132

e	133	x	152	ü	171	^	190		209		228		247
f	134	y	153	ö	172		191		210		229		248
g	135	z	154	Ü	173		192		211		230		249
h	136	ä	155	Ä	174		193		212		231		250
i	137		156	Ö	175		194		213		232		251
j	138		157		176		195		214		233		252
k	139		158		177		196		215		234		253
l	140		159		178		197		216		235		254
m	141		160		179	C	198		217		236		255
n	142		161		180		199		218		237		
o	143		162		181	H	200		219		238		
p	144		163		182		201		220		239		
q	145	`	164		183		202		221	SP	240		
r	146	~	165		184		203		222		241		
s	147		166		185		204		223		242		
t	148		167		186		205		224		243		
u	149		168		187		206		225		244		
v	150		169		188		207		226		245		
w	151	β	170		189		208		227		246		

ASCII-Code-Tabelle für den Plotter-Drucker

Für den Plotter-Drucker gilt die ASCII-Code-Tabelle, jedoch mit der Einschränkung, daß zahlreiche Zeichen und Symbole als ASCII-Werte in hexadezimaler Form und in blauer Farbe gedruckt werden.

Mit dem folgenden Programm kann man sich die für den Plotter-Drucker gültigen ASCII-Werte ausdrucken lassen.

Programmliste:

```
10 REM ASCII-WERTE DES PLOTTER-DRUCKERS
20 REM ---------------------------------
30 PRINT/P 19; " ENTSPRICHT "; CHR$(19)
40 PRINT/P 20; " ENTSPRICHT "; CHR$(20)
50 PRINT/P : PRINT/P
60 FOR I = 32 TO 255
70 PRINT/P  I; " ENTSPRICHT "; CHR$(I)
80 NEXT I
```

Ergebnisse eines Programmlaufs:

```
19 ENTSPRICHT ⊞
20 ENTSPRICHT ⊟


32 ENTSPRICHT
33 ENTSPRICHT !
34 ENTSPRICHT "
35 ENTSPRICHT #
36 ENTSPRICHT $
37 ENTSPRICHT %
38 ENTSPRICHT &
39 ENTSPRICHT '
40 ENTSPRICHT (
41 ENTSPRICHT )
42 ENTSPRICHT *
43 ENTSPRICHT +
44 ENTSPRICHT ,
45 ENTSPRICHT -
```

```
46 ENTSPRICHT .
47 ENTSPRICHT /
48 ENTSPRICHT 0
49 ENTSPRICHT 1
50 ENTSPRICHT 2
51 ENTSPRICHT 3
52 ENTSPRICHT 4
53 ENTSPRICHT 5
54 ENTSPRICHT 6
55 ENTSPRICHT 7
56 ENTSPRICHT 8
57 ENTSPRICHT 9
58 ENTSPRICHT :
59 ENTSPRICHT ;
60 ENTSPRICHT <
61 ENTSPRICHT =
62 ENTSPRICHT >
63 ENTSPRICHT ?
```

```
64 ENTSPRICHT @
65 ENTSPRICHT A
66 ENTSPRICHT B
67 ENTSPRICHT C
68 ENTSPRICHT D
69 ENTSPRICHT E
70 ENTSPRICHT F
71 ENTSPRICHT G
72 ENTSPRICHT H
73 ENTSPRICHT I
74 ENTSPRICHT J
75 ENTSPRICHT K
76 ENTSPRICHT L
77 ENTSPRICHT M
78 ENTSPRICHT N
79 ENTSPRICHT O
80 ENTSPRICHT P
81 ENTSPRICHT Q
82 ENTSPRICHT R
83 ENTSPRICHT S
84 ENTSPRICHT T
85 ENTSPRICHT U
86 ENTSPRICHT V
87 ENTSPRICHT W
88 ENTSPRICHT X
89 ENTSPRICHT Y
90 ENTSPRICHT Z
91 ENTSPRICHT [
92 ENTSPRICHT \
93 ENTSPRICHT ]
94 ENTSPRICHT ↑
95 ENTSPRICHT ←
96 ENTSPRICHT 60
97 ENTSPRICHT 61
98 ENTSPRICHT 62
99 ENTSPRICHT 63
100 ENTSPRICHT 64
101 ENTSPRICHT 65
102 ENTSPRICHT 66
103 ENTSPRICHT 67
104 ENTSPRICHT 68
105 ENTSPRICHT 69
106 ENTSPRICHT 6A
107 ENTSPRICHT 6B
108 ENTSPRICHT 6C
109 ENTSPRICHT 6D
110 ENTSPRICHT 6E
111 ENTSPRICHT 6F
112 ENTSPRICHT 70
113 ENTSPRICHT 71
114 ENTSPRICHT 72
115 ENTSPRICHT 73
116 ENTSPRICHT 74
117 ENTSPRICHT 75
118 ENTSPRICHT 76
119 ENTSPRICHT 77
120 ENTSPRICHT 78
121 ENTSPRICHT 79
122 ENTSPRICHT 7A
123 ENTSPRICHT
124 ENTSPRICHT 7C
125 ENTSPRICHT 7D
126 ENTSPRICHT 7E
127 ENTSPRICHT 7F
128 ENTSPRICHT }
129 ENTSPRICHT 81
130 ENTSPRICHT 82
131 ENTSPRICHT 83
132 ENTSPRICHT 84
133 ENTSPRICHT 85
134 ENTSPRICHT 86
135 ENTSPRICHT 87
136 ENTSPRICHT 88
137 ENTSPRICHT 89
138 ENTSPRICHT 8A
139 ENTSPRICHT ^
140 ENTSPRICHT 8C
141 ENTSPRICHT 8D
142 ENTSPRICHT 8E
143 ENTSPRICHT 8F
144 ENTSPRICHT 90
145 ENTSPRICHT 91
146 ENTSPRICHT e
147 ENTSPRICHT `
148 ENTSPRICHT ~
149 ENTSPRICHT 95
150 ENTSPRICHT t
151 ENTSPRICHT 9
```

152 ENTSPRICHT h
153 ENTSPRICHT 99
154 ENTSPRICHT b
155 ENTSPRICHT x
156 ENTSPRICHT d
157 ENTSPRICHT r
158 ENTSPRICHT p
159 ENTSPRICHT c
160 ENTSPRICHT q
161 ENTSPRICHT a
162 ENTSPRICHT z
163 ENTSPRICHT w
164 ENTSPRICHT s
165 ENTSPRICHT u
166 ENTSPRICHT i
167 ENTSPRICHT A7
168 ENTSPRICHT ö
169 ENTSPRICHT k
170 ENTSPRICHT f
171 ENTSPRICHT v
172 ENTSPRICHT AC
173 ENTSPRICHT ü
174 ENTSPRICHT ß
175 ENTSPRICHT j
176 ENTSPRICHT n
177 ENTSPRICHT B1
178 ENTSPRICHT ü
179 ENTSPRICHT m
180 ENTSPRICHT B4
181 ENTSPRICHT B5
182 ENTSPRICHT B6
183 ENTSPRICHT o
184 ENTSPRICHT l
185 ENTSPRICHT Ä
186 ENTSPRICHT ö
187 ENTSPRICHT ä
188 ENTSPRICHT BC
189 ENTSPRICHT y
190 ENTSPRICHT ‹
191 ENTSPRICHT BF
192 ENTSPRICHT C0
191 ENTSPRICHT BF
192 ENTSPRICHT C0
193 ENTSPRICHT C1

194 ENTSPRICHT C2
195 ENTSPRICHT C3
196 ENTSPRICHT C4
197 ENTSPRICHT C5
198 ENTSPRICHT →
199 ENTSPRICHT C7
200 ENTSPRICHT C8
201 ENTSPRICHT C9
202 ENTSPRICHT CA
203 ENTSPRICHT CB
204 ENTSPRICHT CC
205 ENTSPRICHT CD
206 ENTSPRICHT CE
207 ENTSPRICHT _
208 ENTSPRICHT D0
209 ENTSPRICHT D1
210 ENTSPRICHT D2
211 ENTSPRICHT D3
212 ENTSPRICHT D4
213 ENTSPRICHT D5
214 ENTSPRICHT D6
215 ENTSPRICHT ‾
216 ENTSPRICHT D8
217 ENTSPRICHT D9
218 ENTSPRICHT DA
219 ENTSPRICHT DB
220 ENTSPRICHT DC
221 ENTSPRICHT DD
222 ENTSPRICHT DE
223 ENTSPRICHT DF
224 ENTSPRICHT E0
225 ENTSPRICHT E1
226 ENTSPRICHT E2
227 ENTSPRICHT E3
228 ENTSPRICHT E4
229 ENTSPRICHT E5
230 ENTSPRICHT E6
231 ENTSPRICHT E7
232 ENTSPRICHT E8
233 ENTSPRICHT E9
234 ENTSPRICHT EA
235 ENTSPRICHT EB
236 ENTSPRICHT EC
237 ENTSPRICHT ED

```
238 ENTSPRICHT EE
239 ENTSPRICHT EF
240 ENTSPRICHT F0
241 ENTSPRICHT F1
242 ENTSPRICHT F2
243 ENTSPRICHT F3
244 ENTSPRICHT F4
245 ENTSPRICHT F5
246 ENTSPRICHT F6
247 ENTSPRICHT F7
248 ENTSPRICHT F8
249 ENTSPRICHT F9
250 ENTSPRICHT FA
251 ENTSPRICHT £
252 ENTSPRICHT ↓
253 ENTSPRICHT FD
254 ENTSPRICHT FE
255 ENTSPRICHT π
```

Systemfehlermeldungen

Bei Ausführung des Programms oder eines Kommandos erkennt der BASIC-Interpreter formale Fehler und gibt eine Meldung aus, deren Text aber nicht in jedem Fall direkte Rückschlüsse auf die Art des Verstoßes zuläßt.

Die folgenden Systemfehlermeldungen sind alphabetisch geordnet.

Already open error

Übersetzung: Fehler, da (Datei) schon geöffnet (ist)

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

43

Erläuterungen: Der OPEN-Befehl wurde für eine Datei angewendet, die bereits geöffnet ist.

Array def error

Übersetzung: Fehler wegen (unzulässiger) Felddefinition

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

7

Erläuterungen: Es wurde versucht, ein mehrdimensionales Feld durch erneute Definition mit DIM zu vergrößern. Wenn unzulässigerweise versucht wird, ein eindimensionales Feld durch erneute Definition zu erweitern, erscheint die Fehlermeldung "Illegal data error".

Systemfehlermeldungen:

Can't continue

Übersetzung: Fortsetzung nicht möglich

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)
17

Erläuterungen: Das CONT-Kommando kann nicht ausgeführt werden.

Can't RESUME error

Übersetzung: Fehler, weil RESUME nicht ausgeführt werden kann

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)
20

Erläuterungen: RESUME kann nicht ausgeführt werden, wenn kein Fehler vorliegt.

Check sum error

Übersetzung: Kontrollsummenfehler

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)
70

Erläuterungen: Beim Lesen einer Datei auf Kassettenband trat ein Lesefehler auf.

DEF FN nesting error

Übersetzung: Fehler wegen (unzulässiger) Verschachtelung(stiefe) mit DEF FN

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)
12

Erläuterungen: Die Verschachtelungstiefe bei DEF FN-Anweisungen überschreitet die zulässige Zahl.

Systemfehlermeldungen:

FOR ... NEXT error

Übersetzung: FOR ... NEXT-Fehler

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

11

Erläuterungen: Die Anzahl der FOR...NEXT-Ebenen überschreitet die nutzbare Speicherkapazität.

GOSUB nesting error

Übersetzung: Fehler wegen (unzulässiger) Verschachtelung von GOSUB(-Anweisungen)

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

10

Erläuterungen: Die Verschachtelung von GOSUB-Anweisungen überschreitet die nutzbare Speicherkapazität.

Illegal data error

Übersetzung: Fehler wegen unzulässiger Daten

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

3

Erläuterungen: Es wurde eine unzulässige Konstante oder Variable verwendet.

Instruction error

Übersetzung: Anweisungsfehler

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

19

Systemfehlermeldungen:

	Erläuterungen: Befehle im direkten Modus und Programmbefehle wurden in unzulässiger Weise vermischt.
Line length error	Übersetzung: Fehler wegen (unzulässiger) Zeilenlänge
	Interne Fehlermeldungsnummer: (Inhalt der System-Variablen ERN)
	8
	Erläuterungen: Eine Programmzeile ist zu lang.
Memory capacity error	Übersetzung: Fehler wegen (zu geringer) Speicherkapazität
	Interne Fehlermeldungsnummer: (Inhalt der System-Variablen ERN)
	6
	Erläuterungen: Ein Programm ist im Hinblick auf den zur Verfügung stehenden Arbeitsspeicher zu groß.
Memory protection	Übersetzung: (Fehler wegen) Speicherschutzes
	Interne Fehlermeldungsnummer: (Inhalt der System-Variablen ERN)
	18
	Erläuterungen: Es wurde versucht, Daten in den BASIC-Arbeitsbereich zu schreiben.
NEXT error	Übersetzung: Fehler wegen Verwendung von NEXT (ohne FOR)

Systemfehlermeldungen:

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

13

Erläuterungen: Zu einer NEXT-Anweisung fehlt ein vorangehender FOR-Befehl. / Eine Variable in einer NEXT-Anweisung entspricht keiner vorher aufgeführten Variablen einer FOR-Instruktion.

Out of file error

Übersetzung: Fehler, da Datei keine weiteren Daten enthält

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

63

Erläuterungen: Während des Versuchs, Daten aus einer Datei zu lesen, wurde das Datei-Ende erkannt.

Over flow error

Übersetzung: Fehler wegen Überlaufs (Bereichsüberschreitung)

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

2

Erläuterungen: Das Ergebnis einer Berechnung oder ein Wert ist zu groß, um in irgendeinem BASIC-Format dargestellt werden zu können.

Printer is not ready

Übersetzung: Drucker ist nicht betriebsbereit

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

65

Systemfehlermeldungen:

Erläuterungen: Der Drucker ist nicht angeschlossen.

Printer mode error

Übersetzung: Fehler hinsichtlich (des verwendeten) Drucker-Modus

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

68

Erläuterungen: Es wurde versucht, im Text-Modus einen für den Grafik-Modus reservierten Befehl zu verwenden.

READ error

Übersetzung: Fehler wegen (unzulässiger Verwendung von) READ

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

24

Erläuterungen: READ wurde ohne zugehörige DATA-Anweisung verwendet.

RETURN error

Übersetzung: Fehler wegen Verwendung von RETURN (ohne GOSUB)

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

14

Erläuterungen: Zu einer RETURN-Anweisung fehlt der vorangehende GOSUB-Aufruf.

String length error

Übersetzung: Fehler aufgrund (falscher) Länge eines Strings

Systemfehlermeldungen:

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

5

Erläuterungen: Die Länge eines Strings überschreitet 255 Zeichen.

Un def. line num. error

Übersetzung: Fehler wegen nicht definierter (nicht vorhandener) Zeilennummer

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

16

Erläuterungen: Es wird eine nicht vorhandene Zeilennummer angesprochen.

Syntax error

Übersetzung: Syntaxfehler (Schreibfehler)

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

1

Erläuterungen: Eine Zeile enthält einen BASIC-Rechtschreibungsfehler (z. B. einen unpaarigen Klammerausdruck, einen falsch geschriebenen Befehl, ein unkorrekt geschriebenes Systemkommando, fehlerhafte Interpunktion).

Un def. function error

Übersetzung: Fehler wegen nicht definierter Funktion

Interne Fehlermeldungsnummer:
(Inhalt der System-Variablen ERN)

15

Erläuterungen: Eine nicht definierte Funktion wurde aufgerufen.

Abkürzungen für BASIC-Schlüsselwörter

Der SHARP MZ-700 gestattet eine zeitsparende Eingabe für einige wichtige BASIC-Schlüsselwörter (Operationsteile der Befehle) durch Verwendung von Abkürzungen. Nach Erteilung eines LIST-Kommandos werden sie auf dem Bildschirm aber in Normalform ausgegeben.

Schlüsselwort	Abkürzung	Schlüsselwort	Abkürzung
AUTO	A.	LOAD	LO.
AXIS	AX.	MERGE	ME.
BYE	B.	MID$	MI.
CHR$	CH.	MODE GR	M. GR
CIRCLE	CI.	MODE TL	M. TL
CLOSE	CLO.	MODE TN	M. TN
COLOR	COL.	MODE TS	M. TS
CONSOLE	CONS.	MUSIC	MU.
CONT	C.	ON ERROR GOTO	ON ERR. G.
CURSOR	CU.	ON .. GOSUB	ON .. GOS.
DELETE	D.	ON .. GOTO	ON .. G.
END	E.	PAGE	PA.
FOR ..	F. ..	PCOLOR	PC.
NEXT	N.	PEEK	PE.
GOSUB ..	GOS. ..	PHOME	PH.
RETURN	RET.	PLOT OFF	PL. OFF
GOTO	G.	PLOT ON	PL. ON
GPRINT	GP.	POKE	PO.
HSET	H.	PRINT	?
IF ..	IF ..	PRINT USING	? USI.
GOSUB	GOS.	PRINT/P	?/P
IF ..	IF ..	PRINT/T	?/T
GOTO	G.	PRINT [ZF,HF]	? [ZF,HF]
IF ..	IF ..	READ ..	REA. ..
THEN	TH.	DATA	DA.
INPUT	I.	RENUM	REN.
INPUT/T	I./T	RESTORE	RES.
KEY LIST	K. L.	RESUME	RESU.
LEFT$	LEF.	RIGHT$	RI.
LIMIT	LIM.	RLINE	RL.
LIST	L.	RMOVE	RM.
LIST/P	L./P	ROPEN	RO.

Schlüsselwort	Abkürzung	Schlüsselwort	Abkürzung
RUN	R.	TROFF	TROF.
SAVE	SA.	TRON	T.
SKIP	SK.	USR	U.
STOP	S.	VERIFY	V.
TEMPO	TEM.	WOPEN	W.
TEST	TE.		

Sprachliche Erläuterungen zu den Schlüsselwörtern

(Für die Erläuterung der Aussprache der englischen Wörter haben wir bewußt darauf verzichtet, eine phonetische Umschrift zu verwenden, da deren Kenntnis nicht vorausgesetzt werden kann. Statt dessen verwenden wir eine "Umschreibung", die die Aussprache zwar nicht immer ganz genau wiedergibt, dafür aber ohne weiteres verstanden wird. Das Apostroph (') kennzeichnet die betonte Silbe.)

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
ABS	absolute value	'äb-so-'luut 'wäl-ju	Absolutwert
ASC	American Standard Code for Information Interchange	e-'mä-rie-känn 'stän-dard cood for in-for-'mee-schen 'in-ter-tscheensch	Amerikanischer Standard-Code für Informationsaustausch
ATN	arctangent	'aak-tän-dschent	Arkustangens
AUTO	auto	'o-tou	selbsttätig
AXIS	axis	'ä-xis	Achse(nkreuz)
BYE	bye-bye	bei-bei	Wiedersehen!
CHR$	chracter string	kä-rick-ter string	Zeichenkette
CIRCLE	circle	'ssö-kel	Kreis
CLOSE	to close	to klooss	schließen

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
CLR	to clear	to klier	säubern, reinigen
	Achtung: Die Wirkung des Befehls CLR (löscht u. a. Variableninhalte) darf nicht verwechselt werden mit der Funktion der Taste [CLR] (Stirnseite der [INST]-Taste, also erreichbar durch gleichzeitige Betätigung von [SHIFT] und [CLR]). Die Betätigung der Tasten entspricht der Wirkung des CLS-Befehls (Clear screen).		
CLS	to clear the screen	to klier se skrien	den Bildschirm säubern
COLOR	color (am. engl.)	'kal-ler	Farbe
CONSOLE	console	kon-'ssoul	hier: Bedienungsplatz
CONT	to continue	kon-'tinn-ju	fortfahren, fortsetzen
COS	cosine	'ko-ssein	Kosinus
CURSOR	cursor	'körs-ser	Schieber, Blinkzeichen
DEF FN	to define a function	to die-'fein a 'funk-schen	eine Funktion definieren
DEF KEY	to define a key	to die-'fein a kie	eine Taste definieren
DELETE	to delete	to die-'liet	tilgen, löschen
DIM	dimension	di-'men-schen	Dimension
END	end	end	Ende
ERL	error line	'är-ror lein	Fehlerzeile
ERN	error number	'är-ror 'namm-ber	Fehlernummer
ERROR	error	'är-ror	Fehler

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
EXP	exponentiation	ex-po-nenn-zie-'ee-schen	Exponentiation
FOR .. NEXT	for .. next	for .. näkst	für .. nächstes
GET	to get	to gett	holen
GOSUB .. RETURN	to go to a subprogram / to return	go-'ssapp bzw. to go to a 'ssapp-'pro-grämm / to rie-'törn	zum Unterprogramm gehen / zurückkehren
GOTO	to go to	to go to	gehen nach
GPRINT	graphic / to print	'gräf-fick / to print	grafisch / drucken
HSET	home / to set	hoom / ssett	hier: Ursprung / festlegen
IF ..	if	iff	wenn, falls
IF .. THEN	if .. then	iff senn	falls .. dann
INP	input	'inn-putt	Eingabe
INPUT	input	'inn-putt	Eingabe
INPUT/T	input from tape	'inn-putt fromm teep	Eingabe von Band
INT	integer	'inn-ti-dscher	Ganzzahl
KEY LIST	key list	kie list	Tastenliste
LEFT$	left / string	lefft string	links / String
LEN	length	längss	Länge
LET	to let	to lett	lassen
LIMIT	limit	'lim-mitt	Grenze, Schranke

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
LINE	line	lein	Linie, Strich
LIST	to list	to list	listen, auflisten
LIST/P	to list on the printer	to list onn se 'prinn-ter	auf dem Drucker auflisten
LN	logarithm, natural	lo-ge-'ri-sm 'nä-tschu-rel	Logarithmus, natürlicher
LOAD	to load	to lood	laden
LOG	logarithm	lo-ge-'ri-sm	Logarithmus
MERGE	to merge	to mördsch	verschmelzen
MID$	mid / string	midd string	Mitte / String
MODE GR	mode, graphic	mood 'gräf-fick	Betriebsart (Modus), grafische (grafischer)
MODE TL	mode, text large	mood text laadsch	Betriebsart (Modus), Text groß
MODE TN	mode, text normal	mood text normel	Betriebsart (Modus), Text normal
MODE TS	mode, text small	mood text smoll	Betriebsart (Modus), Text klein
MOVE	to move	to muuv	bewegen
MUSIC	music	'mju-ssick	Musik
NEW	new	nju	neu

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
NEXT	next	näkst	nächstes
ON ERROR GOTO	on error go to	onn 'är-ror go to	im Fehlerfalle gehe nach
ON .. GOSUB	on .. go to subprogram	onn go to 'ssapp-'pro-grämm	abhängig von .. gehe zum Unterprogramm
ON .. GOTO	on .. go to	onn go to	abhängig von .. gehe nach
OUT	output	'aut-putt	Ausgabe
PAGE	page	peedsch	Seite
PAI	enthält die Buchstaben der mathematischen Konstanten PI		
PCOLOR	printer color od. pen color	'prinn-ter 'kal-ler penn 'kal-ler	Druckerfarbe Minenfarbe
PEEK	to peek	to piek	gucken
PHOME	pen to home position	penn to hoom po-'si-schen	Mine zurück zur Ausgangsposition
PLOT OFF	plot (am. engl.) off	plott off	grafische Darstellung, aus(geschaltet)
PLOT ON	plot (am. engl.) on	plott onn	grafische Darstellung, an(geschaltet)
POKE	to poke	to pook	hier: hineinstoßen, "hineinhauen"
PRINT	to print	to print	drucken

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
PRINT/P	to print on the printer	to print onn se 'prin-ter	auf dem Drucker drucken
PRINT/T	to print on the tape	to print onn se teep	auf Band ausgeben (drucken)
PRINT USING	to print using	to print 'ju-sing	drucken unter Benutzung von
RAD	radian	'ree-die-en	Radiant (Bogenmaß)
READ .. DATA	to read / data	to ried / 'dee-ta	lesen / Daten
REM	remark	rie-'maak	Bemerkung
RENUM	to renumber	to rie-'namm-ber	renumerieren
RESET	to reset	to rie-'ssett	zurücksetzen
RESTORE	to restore	to rie-'stor	wiederherstellen, zurückstellen
RESUME	to resume	to rie-'sjuum	wiederaufnehmen, fortsetzen
RETURN	to return	to rie-'törn	zurückkehren
RIGHT$	right / string	reiht / string	rechts / String
RLINE	relative line	'rel-la-tiff lein	relativer Strich
RMOVE	relative move	'rel-la-tiff muuv	relative Bewegung
RND	random	'ränn-demm	zufällig
ROPEN	open for reading	'oo-pen for 'rie-ding	eröffne zum Lesen

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
RUN	to run	to rann	laufen
SAVE	to save	to sseef	sichern
SET	to set	to ssett	setzen, festlegen
SGN	sign	ssein	Vorzeichen
SIN	sine	ssein	Sinus
SIZE	size	sseiss	Größe
SKIP	to skip	to skipp	springen, überschlagen
SPC	space	speess	Leerstelle
SQR	square root	squeer ruut	Quadratwurzel
STOP	to stop	to stopp	anhalten
STR$	string / string	string string	String / String
TAB	to tabulate	to 'tä-bju-leet	tabellarisieren
TAN	tangent	'tänn-dschent	Tangens
TEMPO	tempo	'temm-po	Tempo, Geschwindigkeit
TEST	test	test	Test
TI$	time string	teim string	Zeit / String
TROFF	trace off	treess off	Ablaufverfolger aus(geschaltet)
TRON	trace on	treess on	Ablaufverfolger an(geschaltet)

Schlüsselwort	Sprachliche Herkunft	Aussprache	Übersetzung
USR	user (machine language routine)	'juu-ser (ma-'schien 'läng-gwitsch ruu-'tien)	Benutzer(routine in Maschinensprache)
VAL	value	'well-juu	Wert
VERIFY	to verify	to 'wer-ri-fei	verifizieren, auf Richtigkeit prüfen
WOPEN	open for writing	'oo-pen for 'wrei-ting	eröffne zum Schreiben

Erstellung einer Sicherungskopie von der BASIC-Bandkassette

Wir empfehlen nachdrücklich, von der Original-BASIC-Bandkassette eine Kopie anzufertigen und hiermit zu arbeiten, während die Original-Kassette an einem geschützten Platz aufgehoben wird.

Zur Anfertigung einer Kopie benötigt man eine zusätzliche, handelsübliche Standard-Bandkassette (Fe / C60 oder kürzer !).

V o r g e h e n s w e i s e :

(1) Anschalten des Rechners

Das System befindet sich auf der sog. "Monitor-Ebene", und es erscheint folgende Nachricht auf dem Bildschirm:

```
**  MONITOR 1Z-013A  **
*
```

(2) Ein Teil des Arbeitsspeichers muß jetzt für unsere Zwecke mit Hilfe des Monitor-Kommandos

M (von memory correction)

verändert werden, und zwar folgendermaßen:

*MCF00 — MCF00 schreiben und anschließend [CR] betätigen!
A c h t u n g : MCF00 — Nullen (n i c h t Buchstaben!)

Anschließend ist auf dem Bildschirm folgendes Bild:

```
**  MONITOR 1Z-013A  **
*MCF00
CF00 FF
```

(Das bedeutet: Die Speicherstelle CF00 hat den Inhalt FF - es ist aber möglich, daß auch ein anderer Inhalt angegeben wird.)

Wir schreiben CD und betätigen die [CR]-Taste. Auf dem Bildschirm ist folgender Text:

```
**  MONITOR 1Z-013A  *
*MCF00
CF00 FF CD
CF01 00 27
CF02 FF 00
CF03 00 38
CF04 FF 03
CF05 00 CD
CF06 FF 2A
CF07 00 00
CF08 FF DA
CF09 00 FE
CF0A FF 00
CF0B 00 C3
CF0C FF AD
CF0D 00 00
CF0E FF CD
CF0F 00 27
CF10 FF 00
CF11 00 38
CF12 FF F5
CF13 00 C3
CF14 FF CB
CF15 00 0F
CF16
```

Diese Werte schreiben und mit [CR] abschließen. (0 ist immer eine Ziffer!)



(Es ist nicht unwahrscheinlich, daß das System hier andere Speicherinhalte angibt, was jedoch unerheblich ist.)

Es sind nun die Tasten [SHIFT] und [BREAK] gleichzeitig zu betätigen.

(3) Wir öffnen das Bandkassettenlaufwerk durch Betätigung von [STOP/EJECT] und legen die BASIC-Kassette dergestalt in den Deckel des Laufwerks, daß das Bandmaterial auf der linken Spulenseite liegt. (Die Schreib-/Leseöffnung der Kassette befindet sich am vorderen Rand des Deckels!) Wir schließen den Deckel.

(4) Das Monitor-Kommando J (Jump) ist wie folgt einzugeben:

```
*JCF00          ([CR] drücken!)
⭳ PLAY          (Antwort des Systems)
```

(5) Aufgrund der Systemnachricht (⭳ PLAY) drücken wir auf die [PLAY]-Taste des Bandkassettenlaufwerks.

(6) Das Laufwerk beginnt zu laufen, und der Inhalt der BASIC-Bandkassette wird in den Arbeitsspeicher geladen, ohne daß der Rechner eine diesbezügliche Nachricht ausgibt. (Nur wenn eine ERROR-Meldung erfolgt, ist der Vorgang ab Pkt. (1) zu wiederholen!)

(7) Der Ladevorgang ist beendet, wenn das Laufwerk anhält und ein Stern (*) in der folgenden Zeile erscheint:

```
*JCF00
⭳PLAY
*▒      ⇦
```

(8) Wir betätigen [STOP/EJECT] und spulen das Band durch Drücken von [REWIND] zurück. Wenn das Laufwerk stoppt, betätigen wir wieder [STOP/EJECT] und dann nochmals [STOP/EJECT]. Dadurch öffnet sich der Deckel des Laufwerks, so daß wir die Original-BASIC-Bandkassette gegen eine andere Kassette austauschen können, auf die die Kopie geschrieben werden soll.

(9) Wir schließen den Deckel und erteilen das Monitor-Kommando

```
*J1108          ([CR] drücken!)
```

Es erscheint folgender Text:

```
S-BASICEX SAVER 05▒16▒
HIT ANY KEY?▒
```

(10) Wir drücken eine beliebige Taste, z. B. [L], und das System gibt diese Nachricht aus:

```
⭳RECORD.PLAY
```

(11) Daraufhin drücken wir die Tasten [RECORD] und [PLAY] am Laufwerk (es reicht auch aus, nur [RECORD] zu betätigen, weil [PLAY] dann automatisch niedergedrückt wird).

(12) Das System quittiert dies mit der Nachricht

```
WRITING S-BASIC        .
```

(13) Nach dem Ertönen eines akustischen Signals ist der Kopiervorgang beendet, und es erscheint die Systemnachricht

```
HIT ANY KEY?
```

(14) Wir betätigen den RESET-Knopf, der sich auf der hinteren Buchsen-Seite (Kabeleingangsseite) befindet. Der Monitor meldet sich wieder wie gewohnt:

```
**  MONITOR 1Z-013A  **
*▒
```

(15) Wir betätigen [STOP/EJECT], dann [REWIND], anschließend wieder [STOP/EJECT] und erteilen dann das Monitor-Kommando JCF0E:

```
**  MONITOR 1Z-013A  **
*JCF0E                        ([CR] drücken!)
```

mit der Folge, daß das System uns auffordert, [PLAY] zu betätigen:

```
⭳ PLAY
```

(16) Nach der Betätigung von [PLAY] wird die Bandaufzeichnung auf Richtigkeit überprüft ("VERIFY"-Funktion). Wenn die Aufzeichnung fehlerfrei ist, meldet das System schließlich

```
⭳ PLAYOK!
*
```

Wir drücken [STOP/EJECT], anschließend [REWIND] und dann [STOP/EJECT] und besitzen nun eine einwandfreie Kopie.

(17) Gibt das System hingegen eine ERROR-Meldung aus (z. B. PLAYCHECK SUM ER.), dann ist der beschriebene Vorgang zu wiederholen, und zwar möglichst mit einer anderen Bandkassette.

Literaturhinweise

Wir geben hier bewußt nur eine sehr kleine Auswahl von solchen BASIC-Büchern wieder, die gegenüber anderen Veröffentlichungen einen wesentlichen Zusatznutzen bieten oder Spezialprobleme behandeln.

(1) Ackermann, Hanns:

BASIC in der medizinischen Statistik (Skriptum für Mediziner, Biologen, Pharmazeuten ab 1. Semester), in der Reihe: uni-text, Friedr. Vieweg & Sohn Verlagsges. mbH, Braunschweig 1977

Anm.: Es handelt sich um eine sehr knappe Einführung in BASIC mit Anwendungsbeispielen aus der Statistik im Hinblick auf medizinische / pharmazeutische Untersuchungen.

(2) Ahl, David (Hrsg.):

BASIC Computer Spiele - Band 1 - 101 fantastische Spiele für Ihren Mikrocomputer in BASIC geschrieben mit Listing und Probelauf, SYBEX-Verlag GmbH, Düsseldorf 1982

Anm.: Wegen des oben wiedergegebenen, ausführlichen Untertitels bedarf das Buch keiner weiteren Erläuterungen.

(3) Ahl, David (Hrsg.):

BASIC Computer Spiele - Band 2 - . . . weitere 84 herrliche Computerspiele für Ihren Mikrocomputer - alle in BASIC mit Programm-Listing und Probelauf, SYBEX-Verlag GmbH, Düsseldorf 1982

Anm.: Wegen des oben wiedergegebenen, ausführlichen Untertitels bedarf das Buch keiner weiteren Erläuterungen.

(4) Baumann, Rüdeger:

Computerspiele und Knobeleien programmiert in BASIC, in der Reihe: CHIP Wissen, 3. Auflage, Vogel-Buchverlag, Würburg 1983

Anm.: Das Buch ist eine sehr gute Einführung zur Erstellung von Spielprogrammen in BASIC. Die zahlreichen, gut dokumentierten Beispiele sind unterteilt in "Lernspiele", "Graphikspiele", "Such- und Ratespiele", "Glücksspiele", "Strategiespiele", "Gemischte Strategien", "Geduldspiele" und "Vermischte Knobeleien".

(5) Gilder, Jules H.:

BASIC Computer Programs in Science and Engineering, Hayden Book Company, Inc., Rochelle Park, New Jersey 1980

Anm.: Das Buch enthält 114 sehr gut erläuterte BASIC-Programme, vor allem aus dem Gebiet der Mathematik und den Ingenieurwissenschaften / Schwerpunkt Elektronik: (1) General Mathematics (18 Programme), (2) Engineering Mathematics (6 Programme), (3) Complex Mathematics (8 Programme), (4) Matrix Mathematics (8 Programme), (5) Data Analysis (9 Programme), (6) Basic Electricity (9 Programme), (7) Basic Electronics (7 Programme), (8) Computer-Aided Circuit Design (7 Programme), (9) Active Filter Design (8 Programme), (10) Communications (9 Programme), (11) Passive Filters (14 Programme), (12) Attenuator Pads (11 Programme). - Der Autor verwendete eine BASIC-Version von Microsoft und verzichtete bewußt auf jene Befehle und Funktionen, die wenig verbreitet sind.

(6) Hamann, Günter O.:

Datenverarbeitung mit BASIC, 4. Aufl., Deutscher Betriebswirte-Verlag GmbH, Gernsbach 1983

Anm.: In dem als **P**rogrammierte **U**nterweisung konzipierten Buch beschreiben wir eine sehr verbreitete BASIC-Version (BASIC-80 / Disk und BASIC-86 / Disk von Microsoft), die bei sehr vielen größeren Mikro-Computern verwendet wird.

(7) IBM (Hrsg.):

BASIC - Handbuch für den Informatik-Unterricht in Schulen, IBM Deutschland GmbH, IMB-Form F 12-1030-2

Anm.: Es werden zahlreiche kurze, sehr gut dokumentierte BASIC-Programmbeispiele für den Mathematikunterricht der Sekundarstufe und für wirtschaftswissenschaftlichen Einführungsunterricht wiedergegeben.

(8) Krizan, Peter / Kaufmann, Klaus-Dieter:

Spaß mit BASIC für Anwender - Ein nützliches Programm-Potpourri für alle großen und kleinen Programmierer, Computerfans und Hobby-Computer-Besitzer, IDEA Verlag GmbH, Puchheim 1982

Anm.: Das Buch enthält Programme zu folgenden Themenkreisen: (1) Mathematisch-technischer Themenkreis, (2) Allgemeiner Themenkreis, (3) Themenkreis Wirtschaft, (4) Graphik, (5) Testprogramme (Benchmark-Programme), (6) Spielprogramme, (7) Sprache, (8) Lernprogramme.

(9) Lien, David A.:

The BASIC Handbook - Encyclopedia of the BASIC Computer Language, 2nd edition, Compusoft Publishing, Division of CompuSoft, Inc., San Diego, California 921 119, U. S. A., 1981

Anm.: Das Buch enthält in alphabetischer Ordnung Wirkungsbeschreibungen (mit den evtl. vorhandenen Divergenzen bei verschiedenen Rechnern) zu sehr vielen BASIC-Befehlen. Zu jeder Anweisung ist ein kurzes Testprogramm aufgeführt. Außerdem werden mögliche Alternativen nachgewiesen, falls ein konkreter Rechner den erläuterten Befehl nicht kennt. So weit wir sehen können, handelt es sich um das beste Werk dieser Art.

(10) Menzel, Klaus:

BASIC in 100 Beispielen, 2. Aufl., B. G. Teubner Verlag, Stuttgart 1982

Anm.: Das Buch bietet neben einer äußerst knappen Einführung in die Programmiersprache BASIC hundert kleine Programmbeispiele, untergliedert in die drei Abschnitte "Schulbeispiele", "Spiele und Simulationen" und "Was es noch so gibt".

(11) Nagin, Paul / Ledgard, Henry F.:

BASIC with Style - Programming Proverbs, Hayden Book Company, Inc., Rochelle Park, New Jersey 1978

Anm.: Anhand von neunzehn Maximen (Proverb 1: Don't Panic, Proverb 2: Define the Problem Completely, Proverb 3: Start the Documentation Early, ... Proverb 19: Don't be Afraid to Start Over) erhält der Leser eine Richtschnur für gutes und systematisches Programmieren. Die Verfasser befürworten konsequente "Top-Down-Vorgehensweise" und empfehlen die Befolgung bestimmter Standards, durch die übersichtliche Programme entstehen.

(12) Rosenfelder, Lewis:

BASIC Faster and Better & other Mysteries, Verlag IJG Inc., 1260 West Foothill Blod., Upland, CA 19 786, U. S. A., 1981

Anm.: Das Buch wurde für Benutzer des TRS 80 verfaßt. Es enthält zahlreiche allgemeingültige Programmiertricks, Hinweise und z. T. anspruchsvolle Routinen von allgemeiner Bedeutung, die ohne große Schwierigkeiten auch für den SHARP MZ-700 übernommen werden können.

(13) Tracton, Ken:

The BASIC Cookbook - A complete dictionary of all BASIC statements, commands, and functions - with programming examples and flow charts, TAB BOOKS, Blue Ridge Summit, PA. 17 214, U. S. A., 1978

Anm.: Wenn auch das Buch von David A. Lien (s. o.) als das bessere zu beurteilen ist, kann es doch bisweilen hilfreich sein, wenn man die Möglichkeit hat, zusätzlich das Werk von Ken Tracton zu "konsultieren".

(14) Weber, Karl / Türschmann, Carl Wolfram:

BASIC 1 - Lehr- und Handbuch der Programmiersprache BASIC mit wirtschaftswissenschaftlichen Anwendungsbeispielen, Verlag Paul Haupt, Bern und Stuttgart 1977

Anm.: Die Verfasser beschreiben in Band 1 die BASIC-Elementaranweisungen in formalisierter Sprachdarstellung (Backus-Naur Form), die wissenschaftlichen Anforderungen gerecht wird. Anwendungsbeispiele aus der Betriebswirtschaftslehre: "Exponentielle Glättung", "Critical Path Method (CPM)"; aus der Volkswirtschaftslehre: "Preiselastizität der Nachfrage", "Oszillationsmodell nach J. R. Hicks"; aus der Statistik: "Statistische Maßzahlen", "Lineare Regression und Korrelation".

(15) Weber, Karl / Türschmann, Wolfram:

BASIC 2 - Lehr- und Handbuch der Programmiersprache BASIC mit wirtschaftswissenschaftlichen Anwendungsbeispielen, Verlag Paul Haupt, Bern und Stuttgart 1977

Anm.: Die Autoren erläutern in Band 2 die BASIC-Sonderanweisungen (Matrizen-, File- und Diagnoseanweisungen) in formalisierter Sprachdarstellung (Backus Naur Form), die wissenschaftlichen Anforderungen gerecht wird. Anwendungsbeispiele aus der Statistik: "Matrizenrechnung", "Input-Output-Analyse", "Multiple lineare Regression"; aus der Betriebswirtschaftslehre: "Input-Output-Analyse", "Kostenabweichungsanalyse", "Lineare Programmierung"; aus der Volkswirtschaftslehre: "Input-Output-Analyse", "Multisektorales Spielmodell", "Modifiziertes

Oszillationsmodell nach J. R. Hicks". Ein hervorragendes Kapitel über ältere BASIC-Literatur (vgl. Erscheinungsjahr des Buches) mit umfangreichem Schriftenverzeichnis schließt sich an.

(16) Wissebach, Bertold:

Entscheidungen der Produktionsplanung mit einer Sammlung von BASIC-Programmen, in der Reihe: Interdisziplinäre Systemforschung, Nr. 59, Birkhäuser Verlag, Basel und Stuttgart 1978

Anm.: Das Buch enthält zahlreiche Programmbeispiele aus dem Funktionsbereich der Produktion, insbesondere der Produktionsplanung.

(17) Wittig, Siegmar:

BASIC-Brevier - Systematische Aufgabensammlung, Verlag Heinz Heise GmbH, Hannover 1982

Anm.: Das Buch enthält zahlreiche kleine Beispielprogramme, die systematisch geordnet sind (vor allem zum Zweck der Erläuterung der Funktionsweise verschiedener BASIC-Befehle). Hervorzuheben ist u. a. die übersichtliche Gestaltung.

Notizen:

Notizen:

Notizen:

Notizen:

Stichwortverzeichnis